Eurographics Symposium on Rendering (2006)
Tomas Akenine-Möller and Wolfgang Heidrich (Editors)

Real-time soft shadow mapping by backprojection

Gaël Guennebaud, Loïc Barthe and Mathias Paulin†

IRIT - CNRS - Université Paul Sabatier - Toulouse - France

Figure 1: *A scene including alpha-textured meshes (foliage and wire netting). Left: illustration of realistic soft shadows produced by the average of* 1024 *hard shadows (*2.5*s per frame). Right: our new soft shadow algorithm rendering the same scene at 25 fps without any precomputation.*

Abstract

We present a new real-time soft shadow algorithm using a single shadow map per light source. Therefore, our algorithm is well suited to render both complex and dynamic scenes, and it handles all rasterizable geometries. The key idea of our method is to use the shadow map as a simple and uniform discretized represention of the scene, thus allowing us to generate realistic soft shadows in most cases. In particular it naturally handles occluder fusion. Also, our algorithm deals with rectangular light sources as well as textured light sources with high precision, and it maps well to programmable graphics hardware.

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

1. Introduction

Rendering realistic soft shadows in real-time is a fundamental issue in computer graphics. In addition to increase the realism of rendered images, they simplify the identification of spatial relationships between objects. From the practical point of view, point light sources generate so-called *hard shadows* where a sharp transition is seen between light and umbra. However, because most light sources are extended (area or volume), the intensity of the light smoothly varies from no shadow to full shadow, hence generating *soft shadows* with regions of *penumbra*. While rendering hard shadows only require the computation of the visibility between two points (the shaded point and the light), soft shadows require the complicated evaluation of *how much the light source is visible* from the shaded point, usually expressed as a percentage of visibility.

Targeting real-time non-dedicated soft shadow rendering applications, a well suited algorithm should exhibit the following features:

1. handle dynamic and complex scenes in real-time,
2. be independent of both the receiver's and the occluder's geometry (such as meshes, point clouds, images...),
3. generate shadows as faithful as possible to real ones.

Related work: Recent optimized techniques based on object space silhouette detection provide approximate soft shadows in real-time for reasonably complex scenes (such as penumbra-wedges [AAM03, ADMAM03]). However, in addition to some shortcomings (e.g. wrong occluders fusion), they remain limited to manifold meshes and their complexity increases with the scene complexity, making both the first and the second criteria very difficult to

† e-mail: {guenneba | lbarthe | paulin}@irit.fr

satisfy. On the other hand, shadow maps are image based techniques supporting any type of rasterizable geometry: meshes, point clouds or alpha-textured models (commonly used to represent foliage or wire netting). They are also less sensitive to the scene complexity and for these reasons, we focus on this second family of techniques (note that a recent survey on real-time soft shadows can be found in [HLHS03]).

The shadow map algorithm [Wil78] first renders a depth map of the scene from the light source. Using this depth map, called the *shadow map*, a simple depth comparison determines which pixels of the final image are lit or not. This inexpensive process allows the generation of shadows from scratch at each frame at real-time rate. Unfortunately, shadow maps also exhibit several drawbacks. One is the aliased boundary of hard shadows when the sampling resolution is insufficient. This problem can be solved by increasing the effective shadow map resolution [FFBG01, SD02, MT04, WSP04] or replaced by blur with the percentage closer filtering technique [RSC87]. Another fundamental issue is the limitation to hard shadows and hence, several recent real-time soft shadow methods have been built over the shadow map algorithm. Some require the rendering of multiple shadow maps per light [ARHM00, HBS00], limiting their use to static scenes only. Others, dealing with dynamic scenes, render soft shadows from a single light sample even though this generates well known artifacts because only the object parts visible from the light sample are considered as occluders. However, in addition to this shortcoming, such existing techniques suffer from several other important limitations. For instance, some are limited to planar receivers [SS98] while others improperly 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]. Some hybrid methods [CD03, WH03], based on both shadow maps and silhouette extraction, can only compute a coarse approximation of the external penumbra parts (relatively to the hard shadow boundaries). Image based algorithms still have to be improved in order to provide real-time rendering of dynamic scenes with more realistic soft shadows.

In a very recent work [AHL*06], Atty et al. have presented a soft shadow algorithm whith a percentage of visibility computation based, as in our approach, on the back projection of the shadow map samples. However, unlike our algorithm, their method relies on a very restrictive assumption: occluders and receivers are disjointed sets. Moreover, they are limited to very small shadow map resolution (they report 200×200 pixels) and because all visibility computations are done in the discrete shadow map space, shadows are more aliased.

Contribution: We present a new soft shadow algorithm. Rather than limiting the use of the shadow map to simple depth queries, we consider the shadow map as a discretized representation of the scene, each sample being a small potential occluder. The key idea is to use this simplified scene representation to compute the percentage of visibility between a scene point and the extended light source. In order to provide real-time performance, our algorithm uses a single shadow map per light source. Despite this approximation, in most cases, our algorithm provides realistic soft shadows with almost correct occluder fusion. Moreover, it handles any type of rasterizable geometry, it deals with both rectangular and textured light sources, and it is easy to integrate into existing applications. The real-time performance can be guaranteed by using modular accuracy and finally it does not require any precomputation.

2. Our soft shadow algorithm

Even though our algorithm naturally deals with multiple rectangular light sources and with rectangular shadow map pixels, in order to simplify the explanations, we present the main procedure on a single square light source of width l and on square pixels.

Our algorithm computes for each light source a soft *visibility buffer* (V-buffer) modulating diffuse and specular lighting in the final rendering pass. The visibility buffer stores a *visibility factor* $\nu_p \in [0,1]$ for each pixel of the screen thus providing the percentage ($\nu_p * 100$) of light seen by a single pixel: fully lit if $\nu_p = 1$, in umbra if $\nu_p = 0$ and in penumbra when $0 < \nu_p < 1$.

The critical problem which our method solves is then how to perform a fast and accurate computation of the V-buffer. The algorithm is decomposed into two steps: first, it computes the shadow map from the light and then, for each pixel of the V-buffer, if its corresponding point $\mathbf{p}$ is in the penumbra, its visibility factor ν_p is evaluated by computing the light area occluded by a subset of the shadow map samples. This subset of shadow map samples is called the *search area* and it is denoted as A. The visibility computations are detailed in section 2.1 while optimizations, i.e. the penumbra classification and the selection of the search area A, are described in section 2.2.

2.1. Visibility computation

Shadow map acquisition

The first step of the visibility buffer computation is the acquisition of the shadow map storing linear light space depth values. This step requires the definition of a projection frustum (illustrated in figure 2a) having its origin at the light source center. The *near plane* and its borders are taken parallel to the light. It is at a distance n from the origin and its width is w. The other frustum parameters can be chosen arbitrarily or, better, dynamically adapted to the view point in order to optimize the effective shadow map resolution. Finally, the shadow map resolution r should be a power of two in order to simplify the construction of a hierarchical version (section 2.2).

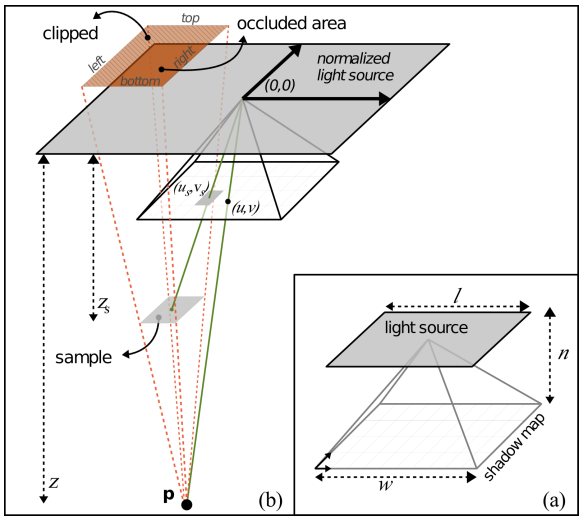

Figure 2: *(a) Shadow map parameters. (b) Projection of a sample onto the light source from the current point* **p**. *This projection is clipped by the light source and the rest is the occluded area.*

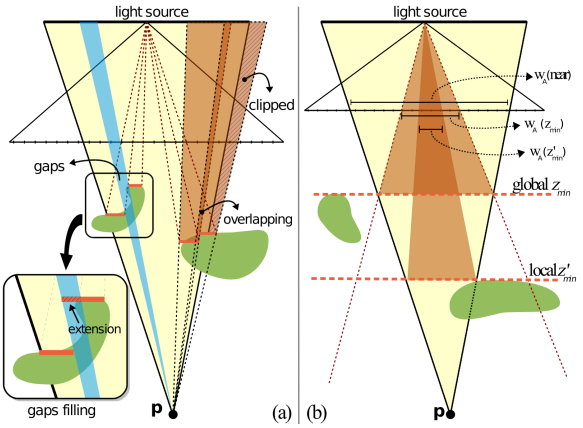

Figure 3: *(a) Illustration of gaps and overlaps. (b) Optimization of the search area.*

Visibility pass

Given the shadow map, the goal of the visibility pass is to compute the visibility factor ν_p of each visible point **p** of the scene. Thus, given a point **p** that has to be shaded, let (u,v) and z be respectively the coordinates (in pixels) of its projection onto the shadow map and its light space depth value (figure 2b). The basic algorithm for computing its visibility factor ν_p is straightforward. We first assume that this point is fully visible ($\nu_p = 1$) and we remove from ν_p the area occluded by every potential occluding sample stored in the shadow map.

The area occluded by a sample of coordinates (u_s, v_s) and depth z_s is computed as follow. Its object space representation, i.e. a square parallel to the light and the shadow map, is projected from the current point **p** onto the light source plane (figure 2b). This projection is a rectangle parallel to the light's borders. Let **B** be its bounds in the normalized light source space, i.e. the two-dimensional light space scaled such that the light size is 1 (figure 2b). Thus B is given by:

$$\mathbf{B} = \begin{bmatrix} b_{left} \\ b_{right} \\ b_{bottom} \\ b_{top} \end{bmatrix} = \begin{matrix} (u_s - u - 1/2) \\ (u_s - u + 1/2) \\ (v_s - v - 1/2) \\ (v_s - v + 1/2) \end{matrix} \quad \frac{w}{n*r} z_s \quad \frac{z}{z - z_s} \quad \frac{1}{l} \qquad (1)$$

where, $\frac{w}{n*r} z_s$ is the size of the sample in the object space and $\frac{z}{z-z_s}$ is the scale between the sample plane and the light source plane. Finally, the intersection between the light and the sample is given by clamping the bounds B by $[-1/2, 1/2]$ and the normalized occluded area $(b_{right} - b_{left}) * (b_{top} - b_{bottom})$ is subtracted from ν_p.

Note that up to now, we consider every sample of the shadow map with a depth value less than z as the set of potential occluders. We show in section 2.2 how to drastically reduce this set of samples (i.e. the search area A).

Gap filling

Owing to the discontinuous representation of the shadow map, small overlaps and gaps may occur between samples (as illustrated in figure 3a). In overlapping regions, some light points are removed twice, hence generating slightly darker penumbrae. On the other hand, in gap regions, occluded light points are not removed and this may create unwanted light in umbra regions (see figure 7d). Whereas overlap errors are quite acceptable, gap artifacts have to be removed. In order to fill the gaps, we make the assumption that they occur between neighbor samples of the shadow map and then we extend samples' left and bottom boundaries such that all occluding samples join in the two-dimensional light space (figure 3a). Whereas the previous assumption is true in 1D, it is not always the case in 2D because samples are shifted in the two directions. This explains why we only extend the samples to fill gaps and why we cannot also clip the samples to remove overlapping. Hence, this procedure slightly increases the overlap error, but it remains rarely perceptible while it effectively fills the gaps. We also point out that the overlap error may becomes large as soon as their exist two occluders such that one is close to the light source and the other is close to the current shaded point. Fortunately, such extrem cases rarely occur in practice. To summarize, for a given occluding sample (u_s, v_s):

- compute its bounds **B** using equation 1,
- compute the bound b'_{left} (resp. b'_{bottom}) for its neighbor $(u_s - 1, v_s)$ (resp. $(u_s, v_s - 1)$) by adapting equation 1,
- take as final value for b_{left} (resp. b_{bottom}) the minimal value between b'_{left} and b_{left} (resp. b'_{bottom} and b_{bottom}).

Obviously, a sample is extended towards one of its neighbors if and only if this neighbor is also a potential occluder, i.e. if its depth is lower than the current receiver depth.

Textured light source

The method presented above uses a fast analytical computation of the area occluded by each sample. This method can easily be adapted to handle textured and animated light sources (like a fire) in a similar way to that in [AAM03]. Indeed, the four scalar values **B**, defining the light region occluded by a sample, can be directly used to index a 4D texture storing the normalized light source color of this region. This way, we can handle any kind of area light source. All details on this method can be found in [AAM03]. The use of 4D textures generates discretization artifacts and the textures are expensive to store. Since our occluding regions are rectangular quads parallel to the light, a dramatically lighter and more accurate alternative is to use Summed Area Tables (SAT) [Cro84]. Textured light sources are illustrated figure 4.

2.2. Optimizations

Significant improvements in performance can be made out by both reducing the number of potential occluding samples and performing the expensive penumbra calculations only on pixels potentially in the penumbra. In practice, the second optimization requires the first one.

In order to implement these optimizations, we first have to build a hierarchical shadow map (HSM) storing for each pixel both minimal and maximal depth values of occluder samples (level 0 is just the shadow map itself). As with mipmaps, other levels are built iteratively by reducing the resolution by a factor two, but rather than average values, each low level pixel stores both the minimal and maximal covered depth values. In practice, our HSM is efficiently stored in a hardware mipmapped texture such that these reduction steps are efficiently performed by the GPU: at each step, the largest level is rendered into the lower one with a trivial fragment shader performing custum minimal and maximal operations.

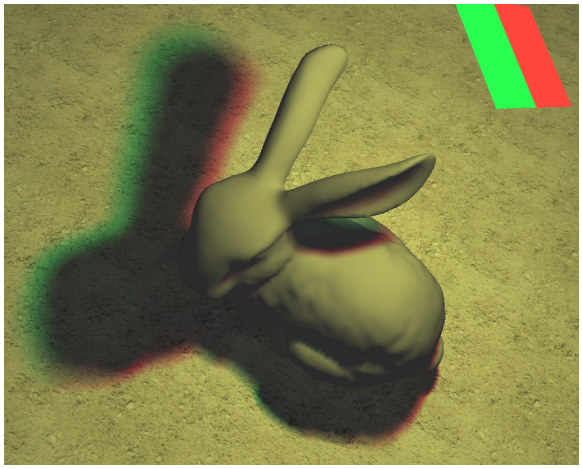

Figure 4: *Soft shadows from a simple textured light source. For this example we have used the SAT option.*

Search area reduction

In order to reduce the number of potential occluding samples, we accurately evaluate the search area A. Since the occluding samples are inside the rectangular pyramid formed by the light quad and the current point **p**, a first coarse approximation is obtained by taking the intersection between the near plane of the shadow map and the pyramid (figure 3b). The subset A is then a square (or a rectangle if the light is rectangular) of width $w_A = l\frac{n}{w}\left(\frac{1}{n}-\frac{1}{z}\right)$, centered in (u,v) (the projection of **p** onto the shadow map). This approximation can be improved in two steps. First, the top level of the HSM gives us the minimal depth value z_{min} stored in the shadow map. Thus, the pyramid can be clamped by a plane of depth z_{min} (figure 3b) and the width of the new search area becomes:

$$w_A = l\frac{n}{w}\left(\frac{1}{z_{min}}-\frac{1}{z}\right) \qquad (2)$$

Then, we quickly find the local min depth value z'_{min} of the new search area (figure 3b) by accessing the appropriate level in the HSM (note that four texture fetches are required). The new value of w_A is computed with equation 2 using z'_{min} rather than z_{min}. This step can be repeated while the search area is significantly reduced, but in practice, we have found that one step is sufficient.

Penumbra classification

We want now to quickly classify potential penumbra pixels in order to reduce the number of expensive computations of visibility factors v_p. For the current point **p**, once the accurate local depth bound values z'_{min} and z'_{max} are evaluated (during the search area reduction), we compare its depth value z:

1. if $z \leq z'_{min}$ then the pixel is fully lit and $\mathrm{v}_p = 1$,
2. if $z > z'_{max}$ then the pixel is considered as occluded and $\mathrm{v}_p = 0$,
3. otherwise the pixel is potentially in penumbra and v_p must be computed accurately.

Adaptive precision

Finally, the speed of our algorithm mainly depends on the size in pixels of the search areas which can be important for a very large penumbra. In terms of visual quality, large penumbrae require less details than thinner ones. Hence, when real-time performance requires faster computation, we reduce the precision of large penumbrae by dynamically selecting, for each point **p**, a level in the HSM so that the

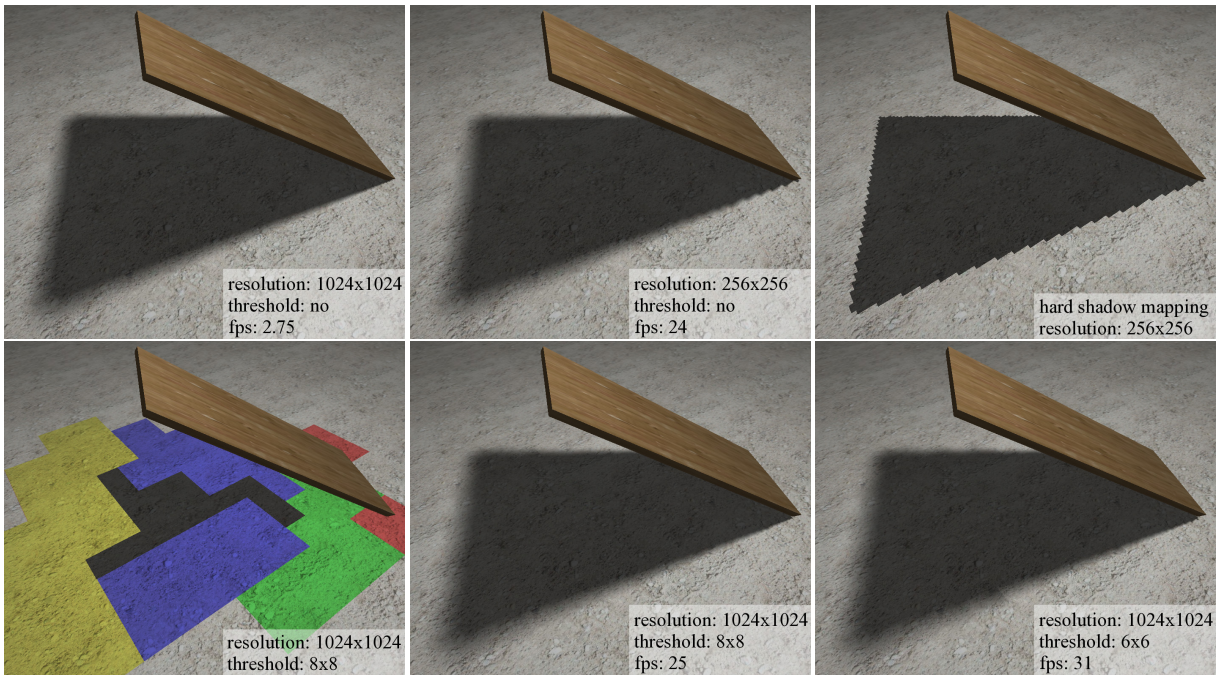

Figure 5: *In these pictures, the frame rate has been measured with a 768×768 screen resolution. Top row: our algorithm without adaptive precision applied with two different resolutions (left and center) compared with standard hard shadow mapping (right). Bottom row: illustration of the adaptive precision capabilities of our algorithm for two different threshold values of the search area (center and right). Colors in the left picture indicate the locally selected level in the HSM. Colors red, green, blue and yellow respectively correspond to levels of resolution 1024×1024, 512×512, 256×256 and 128×128.*

search area does not exceed a given size threshold (figure 5). Therefore, for large penumbrae, rather than backprojecting many samples of the shadow map (level 0 of the HSM), we project fewer but larger samples from a coarser level of the HSM. Since we cannot store any coverage information per pixel of the HSM (coverage values depend on the receiver depth), we simply take as sample depth z_s the minimal depth value in the selected coarse level. When the scene is composed of large penumbra areas, this approximation leads to small visual quality degradations that are described and discussed in section 4.1.

3. Implementation

Our soft shadow mapping algorithm can be implemented in several ways, depending on the application and the hardware. Since there is no general best solution, we present our implementation and discuss the variants.

Deferred shading like strategy

In order to be independent of the scene complexity we use a deferred shading like strategy: at each frame, the scene is rendered a first time from the view point without any shading calculation but, as for the shadow map, into a single component floating point buffer storing linear depth values. In our implementation this depth buffer is only used by the visibility passes to compute the V-buffers (one per light source) and hence the whole geometry is rendered a second time for the final rendering pass (this pass can take advantages of early z-cull). We have opted for this approach rather than a full deferred shading approach because it is more flexible and it provides a better hardware anti-aliasing support.

Dynamic branching versus multi-pass

The simplest way to compute the visibility buffer is to use a single pass with one fragment shader performing the three following steps:

1. compute the reduced search area (using the HSM),
2. classify the pixel as fully occluded, fully lit or in penumbra,
3. if the pixel is in penumbra, loop over samples of the search area and accurately compute the visibility factors.

In this case, efficient dynamic branching support in the fragment shader stages is required to both loop over the selected samples and avoid complex computations on pixels outside the penumbra. However, in some hardware, branching is expensive.

The test “in penumbra?" can be attractively substituted by two passes taking advantage of *early-z* fragment rejection.

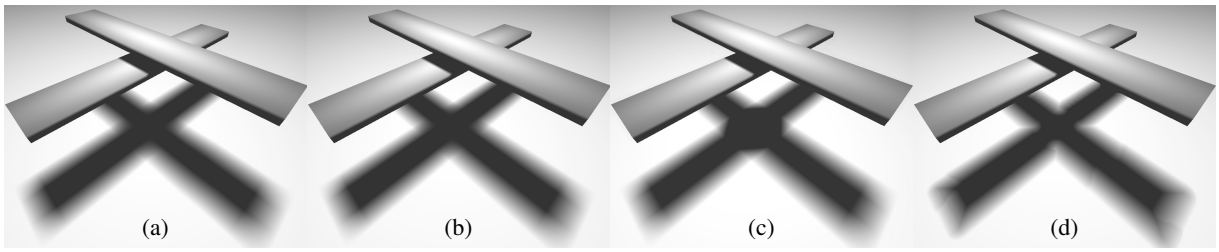

Figure 6: *Our algorithm (a), is compared to a reference image (b), the penumbra wedges technique (c) and the flood fill algorithm (d).*

Basically, for each pixel, the first pass computes the reduced search area and performs the classification (steps 1 and 2). Accordingly, fragment depth is set such that only pixels classified as in penumbra pass the depth test during the next pass (first part of step 3). Then, the dynamic loops can be replaced by static loops if we set the size of the search area. The choice of this size directly depends on the performance one wants to achieve: a small size leads to fast computations but the selection of a coarse level in the HSM to evaluate the visibility factor of large penumbrae (section 2.2) while large size produces accurate soft shadows but requires more expensive computations.

We have tested our algorithm on current NVidia GPUs (GeForce 6x00 and 7x00). Due to the coarse support of dynamic branching of these GPUs (groups of 64×64 fragments follow the same branch), it is difficult to determine the best options since performances can significantly vary according to the scene and the view point position. However, after many experiments and because the dynamic loops treat fewer samples, the two passes approach combined with dynamic loops seems to be the best compromise. We believe that, on GPUs having efficient dynamic branching support, such as the latest ATI GPU generation (X1x00) which manages small groups of 4×4 pixels, the fully dynamic version would be the best choice in any case. Such hardware should also exhibit significantly better relative performances.

Scene	Fig. 7	Fig. 1	Fig. 8
Shadow map	1.7	2.6	8.7
Camera depth map	0.7	1.3	7.6
HSM construction	3.1	3.1	3.1
Visibility pass 1	0.9	0.9	0.9
Visibility pass 2	39	28	15
Final rendering pass	0.8	1.6	8.2
Total (ms)	46.2	37.5	43.5
fps	21.6	26.6	23

Table 1: *Rendering times in ms for each step of our algorithm when it is applied on different scenes (without adaptive resolution).*

4. Results

In this section, we present performances and visual results followed by a short discussion on limitations. The results presented here have been obtained with a GeForce 7800 graphics card.

4.1. Performance

Our algorithm is very efficient, especially for complex dynamic scenes. Indeed, with respect to classical shadow mapping algorithms, our additional visibility calculations do not depend on the scene complexity and they are only performed on visible points which are in or close to the penumbra. For instance, the scene in the figure 8, composed of 800k polygons, is rendered at about 40 fps with classical shadow mapping and at 23 fps using our soft shadow algorithm without adaptive resolution. The rendering times of each part of our algorithm are summarized in the table 1 for scenes of various complexities. These results clearly show that our visibility calculations do not depend on the scene complexity at all, but rather on the size of the penumbrae and the resolution of the shadow map.

In practice, the worst cases are close views on large penumbrae with a high shadow map resolution. In such cases, our performance can drop down to a few frames per second at full precision. For instance, the top left picture in figure 5 shows a simple scene rendered at 2.75 fps. The low performance is explained by the size of the search areas which exceed 32×32 for pixels situated in large penumbrae. This figure also illustrates the effects of our adaptive precision strategy (section 2.2) on both performance and visual quality. As we can see, a threshold value setting the maximal size of the search area at 8×8 is sufficient to successfully meet real-time rates and this with a low visual quality degradation. Indeed, the use of local low resolutions still provides a high quality penumbra except at the transition between different levels of the HSM (see the discontinuities in the penumbra of figure 5 bottom right). Note that, even though taking minimal depth values in the low levels of the HSM may seem to be a coarse approximation (section 2.2), for the same speed, this approximation clearly provides a better visual quality than reducing the resolution of the whole shadow map (figure 5 central images).

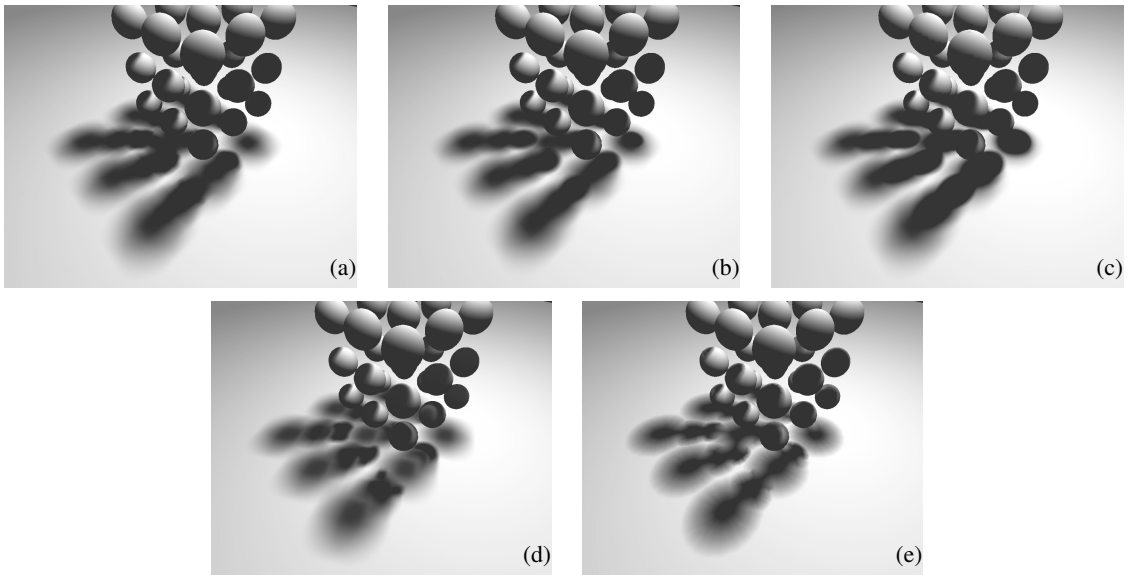

Figure 7: *(a) Our algorithm. (b) Reference image. (c) The penumbra wedges technique. (d) Our algorithm without gaps filling. (e) The flood fill algorithm.*

4.2. Visual results

In order to evaluate our visual results, we take as reference the average of several accurate hard shadows, e.g. high resolution hard shadow maps (2048×2048) computed from 1024 point light samples.

Figure 1 demonstrates the capability of our algorithm to deal with binary alpha-textured meshes (wire netting and foliage) without performance penalty and with the same visual quality as for any other rasterizable geometry. In the figures 6 and 7 we compare our algorithm against the famous penumbra wedges technique [AAM03] and a recent flood fill algorithm [AHT04] on two tedious synthetic examples. As we can see, our algorithm correctly handles occluder fusion when the occluders are close respectively (figure 6), while the other methods do not. When the blend of occluders becomes more complex (figure 7) our algorithm exhibits darker penumbra where the samples of the shadow map overlap too much. However, our results are still closer to the reference than those obtained with other methods. Figure 7d also illustrates the importance of our gaps filling method.

5. Discussion and conclusion

Our algorithm significantly improves the capability of image-based methods to generate dynamic soft shadows in real-time. In addition to providing all the advantages of classical shadow maps (such as independence from the geometry), it produces soft shadows with similar quality to real-time silhouette extraction based techniques but without scalability limitations. Furthermore, our algorithm can handle textured light sources, it is easy to integrate into applications and it does not require any precomputation.

In order to exhibit all these important features, our technique uses some approximations. Since we use a single shadow map, we consider as occluders only object parts that are visible from the center of the light. However, some hidden parts can be visible from other points of the light. Thus these object parts should be treated as occluders whereas they are not considered as such by our method (see figure 9). Artifacts due to this approximation are the same as the *single silhouette artifacts* of the penumbra-wedges technique [AAM03].

Even though these artifacts are seldom perceptible, they can be significantly reduced by splitting the light source area as proposed in [ADMAM03] for the penumbra wedges technique. In this case it is important to notice that, in the case of simple scenes, almost the same performance is obtained when n small light sources are used rather than a single light source of the same area. Indeed, whereas the

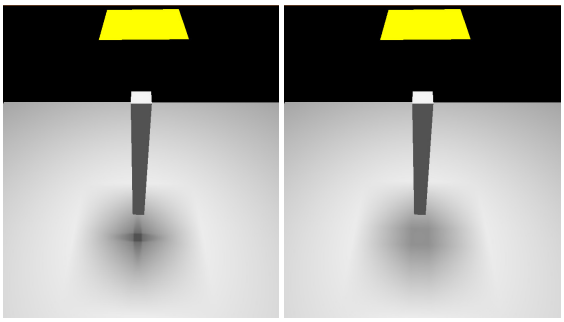

Figure 9: *Illustration of the well known single light sample artifact. Left: reference image. Right: our algorithm.*

Figure 8: *A scene composed of 800k polygons (screen resolution: 768×768 pixels). From left to right: reference image obtained from the average of* 1024 *light samples (*17*s per frame), our algorithm without adaptive resolution (23 fps) and standard shadow mapping with hardware 2×2 percentage closer filtering showing the pixels of the shadow map (40 fps).*

rendering time of the shadow maps will be multiplied by a factor of n, our visibility computations will not increase since they are linear with respect to the light source area.

In the future we intend to increase the accuracy of our soft shadows. This can be achieved by first removing sample overlappings, and then improving the quality of our adaptive precision strategy when the shadow map resolution is too high. Also, when the local shadow map resolution is too low, it becomes necessary to increase its effective resolution. Hence, it could be pertinent to investigate the integration of such existing methods for hard shadow mapping into our soft shadow mapping algorithm.

References

[AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A geometry-based soft shadow volume algorithm using graphics hardware. *Proceedings 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 *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (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* (2006). (to appear).

[AHT04] ARVO J., HIRVIKORPI M., TYYSTJÄRVI J.: Approximate soft shadows using image-space flood-fill algorithm. In *Proceedings of Eurographics 2004, Computer Graphics Forum* (2004), vol. 23, pp. 271–280.

[ARHM00] AGRAWALA M., RAMAMOORTHI R., HEIRICH A., MOLL L.: Efficient image-based methods for rendering soft shadows. In *Proceedings of ACM SIGGRAPH 2000* (2000), ACM Press, pp. 375–384.

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

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

[Cro84] CROW F. C.: Summed-area tables for texture mapping. In *Proceedings of ACM SIGGRAPH '84* (New York, NY, USA, 1984), ACM Press, pp. 207–212.

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

[HBS00] HEIDRICH W., BRABEC S., SEIDEL H.-P.: Soft shadow maps for linear lights high-quality. In *Rendering Techniques '00 (Proceedings of the 11th EG Workshop on Rendering* (2000), Eurographics Association, 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).

[MT04] MARTIN T., TAN T.-S.: Anti-aliasing and continuity with trapezoidal shadow maps. In *Proceedings of the 15th EG Symposium on Rendering* (2004), Eurographics Association.

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

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Proceedings of ACM SIGGRAPH '02* (New York, NY, USA, 2002), ACM Press, pp. 557–562.

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

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

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proceedings of ACM SIGGRAPH '78* (New York, NY, USA, 1978), ACM Press, pp. 270–274.

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