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

Volume 26 (*2007*), *Number 3*

Bitmask Soft Shadows

Michael Schwarz and Marc Stamminger

Computer Graphics Group, University of Erlangen-Nuremberg

Abstract

Recently, several real-time soft shadow algorithms have been introduced which all compute a single shadow map and use its texels to obtain a discrete scene representation. The resulting micropatches are backprojected onto the light source and the light areas occluded by them get accumulated to estimate overall light occlusion. This approach ignores patch overlaps, however, which can lead to objectionable artifacts. In this paper, we propose to determine the visibility of the light source with a bit field where each bit tracks the visibility of a sample point on the light source. This approach not only avoids overlapping-related artifacts but offers a solution to the important occluder fusion problem. Hence, it also becomes possible to correctly incorporate information from multiple depth maps. In addition, a new interpretation of the shadow map data is suggested which often provides superior visual results. Finally, we show how the search area for potential occluders can be reduced substantially.

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

1. Introduction

Shadows are a fundamental ingredient to the realistic appearance of rendered images. Especially in real-time graphics, many algorithms focus on the restricted case of hard shadows cast by point light sources since the underlying point-to-point visibility problem can be solved efficiently and extremely fast. However, in reality, light sources are usually of a certain extent. Hence points within a scene cannot only be either completely lit or entirely in umbra, but also lie in a transition region of partial illumination, the penumbra.

The resulting soft shadows are much harder to deal with since they require determining the fraction of the light source that is visible from a point to be shaded—or at least getting a good estimate of this percentage. While therefore much information is required for accurate results, the amount that can be obtained and processed within a limited time frame is restricted. Consequently, to attain interactive or even real-time frame rates without extensive precomputations or limiting dynamic changes, approximations must be made.

Over time, many interactive and real-time algorithms have been devised which try to provide ever better approximations. Often, further increases in the computational power and programmability of GPUs enabled both the adaptation of CPU-based off-line algorithms as well as completely new approaches, thus helping to improve the achievable soft shadow quality. Recently, a new technique has been introduced which takes the data of a single shadow map as a discrete representation of potential light blockers and uses the resulting micropatches to estimate the light source's visibility. Related algorithms are discussed in more detail in Section 2.1.

In this paper, we extend on their underlying idea of backprojecting micropatches and, making substantial use of latest generation graphics hardware's new features, further improve on the soft shadow quality achievable in real-time. Our contributions include

- a new real-time soft shadow algorithm based on tracking light source visibility via bit fields that correctly performs occluder fusion (Section 3),
- a new interpretation of depth map data that results in improved visual quality (Section 4), and
- a new multi-scale depth map representation that often enables substantial speed-ups (Section 5).

In combination, these contributions usually offer a higher visual quality than related algorithms while attaining comparable frame rates.

2. Related work

An exhaustive treatment of soft shadow algorithms for interactive and real-time rendering is beyond the scope of this paper. We hence concentrate on the most relevant ones for reasonably sized area light sources and refer the reader to a more complete survey [HLHS03] for further information.

One major class of techniques doesn't aim at computing exact soft shadows but tries to fake them in a plausible way, often making crude approximations. Many of them are based on the idea of starting with the hard shadow obtained for a point light source placed at the center of the area light and extending it outwards to generate a penumbra region [PSS98]. Brabec and Seidel [BS02] search a shadow map for the nearest texel containing depth information of an occluder and estimate a pixel's placement within the found occluder's soft shadow. In similar work, Kirsch and Döllner [KD03] move all involved computations to the GPU but become further restricted to handle only inner penumbrae. Arvo et al. [AHT04] create outer penumbrae by applying a modified flood-fill algorithm to the umbra regions obtained from a shadow map. It has also been suggested to simply blur hard shadows via percentage closer filtering [RSC87]. To better account for varying penumbra widths, Fernando [Fer05] employs the result of an occluder search in the shadow map to derive the kernel size for the filtering step. While all these algorithms operate only in image space, some hybrid approaches additionally take object information into account. They render extra geometry attached to silhouette edges to add penumbrae to hard shadow regions [CD03, WH03].

Due to their numerous approximations which enable real-time or at least interactive performance, all algorithms of this class suffer from several inherent limitations which often result in clearly visible artifacts. For instance, because of bounding the occluder search, relevant occluders might get ignored, limiting the penumbra's extent and causing transition artifacts. Moreover, the umbra regions are often significantly overestimated.

To obtain accurate soft shadows, a rather exact knowledge is required about the fraction of the light source's area that is visible from a point to be shaded. Another major class of soft shadow algorithms actually tries to determine this point-to-region visibility fraction quite accurately to derive the amount of shadowing. Some high-quality and computationally expensive methods exist which sample the light source's area, determine the resulting shading-point-to-light-point visibility relations and average the results. In their offline algorithm, Laine and Aila [LA05] loop over all potential blocker triangles to hierarchically determine these point-to-point visibilities. Similar to our approach, they track visibility relations with a bit field for each shading point.

Many other approaches, however, project the occluding geometry onto the light source area and derive the occluded fraction. Usually, interactive algorithms don't consider all potential occluders but restrict themselves to a subset of them. While most of these techniques operate on a single visibility percentage value, in our algorithm we actually use a kind of low resolution binary occlusion image of the light area stored as bit field. This enables the detection and correct handling of overlapping occluder geometry.

Assarsson and Akenine-Möller [AAM03] introduced a soft shadow volume algorithm which renders a penumbra wedge for each silhouette edge (as seen from the center of the light source). A fragment shader projects the edge onto the light source and determines the covered area. These areas are accumulated and used to modulate a visibility buffer initialized by a shadow volume pass. The resulting soft shadows are of rather high quality if the used silhouette edges are roughly identical to those seen from other points on the light source and if the silhouettes don't overlap. To alleviate the latter limitation, Forest et al. [FBP06] suggest to keep track not only of the covered area but also its bounding box for each quarter of the light source. This allows detecting potential overlaps and estimating their magnitude.

Even though projection-based visibility determination has been performed mainly in object space, image space algorithms offer several advantages, especially in rasterization-based interactive computer graphics. They not only scale better with scene complexity but also can readily deal with geometry altered in the fragment stage via alpha masking, fragment kills or depth modifications. Note that especially the selective discarding of fragments is fundamental to many recent techniques for ray-casting surfaces [LB06] and for adapting the silhouette in extended bump mapping algorithms achieving displacement-mapping-like results [OP05].

2.1. Soft shadows via micropatch backprojection

Recently, several algorithms have been introduced which take the depth image obtained from the light source's center as a discrete representation of potential occluders [AHL*06, GBP06, ASK06, BS06], with each depth map texel being considered as a rectangular *micropatch*. To estimate the fraction of the light area visible from a certain point **p**, all relevant micropatches are first unprojected into world space and then backprojected from **p** onto the light's plane. The areas of the intersections of the micropatches' projections and the light area are accumulated to determine the occluded light area. Alternatively, the relative solid angle covered by the micropatches' bounding spheres can be used [BS06] to derive the amount of occlusion.

The actual order of computation as well as the number of considered micropatches varies between the algorithms. Atty et al. [AHL*06] loop over all micropatches and scatter the occlusion caused by them to all affected pixels in a soft shadow map which finally gets projected onto the scene. While this approach is of high efficiency, it requires a separation of objects into shadow casters and shadow receivers. Other approaches don't suffer from this limitation

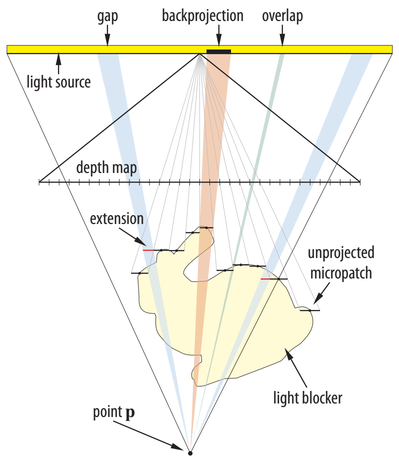

Figure 1: *To determine the visibility of the area light source from a point* **p***, relevant depth map texels are unprojected into world space, the resulting micropatches backprojected onto the light source, and the occluded areas accumulated. While gaps emerging in the reconstruction of light blockers can be treated via micropatch extension, overlapping artifacts are ignored by current techniques.*

since they adopt a gathering strategy where for each point to be shaded all relevant micropatches are fetched from the depth map, backprojected and their area accumulated. Apart from enumerating all depth map texels within a search area determined by the intersection of the point–light-area pyramid and the near plane used for depth map generation [GBP06, ASK06], it has also been suggested to sample according to a Gaussian Poisson distribution [BS06].

While often yielding good visual results and being superior to fake soft shadow techniques, all algorithms of this micropatch backprojection class suffer from several problems, nevertheless. First, other light blockers which cannot be seen from the light source's center are wrongly ignored, which can lead to noticeable artifacts. Note that this problem is somewhat different from the silhouette-from-single-point limitation in the soft shadow volume algorithm where we do consider silhouettes occluded from the light source's center but are then often faced with overlapping silhouettes. Second, the assumption is made that the light area projections of the micropatches don't overlap. While initially recording only occluders visible from the light's center surely helps to avoid overlaps, the assumption often fails nevertheless, sometimes even causing clearly objectionable artifacts. Our new algorithm alleviates both of these problems by being able to account for arbitrary overlaps and hence enabling the inclusion of micropatches from further depth maps.

Third, gaps can occur between neighboring micropatches. Since usually such gaps are undesired holes in surfaces leading to disturbing light leaks and not actual occluder-free regions, it is reasonable to try to close them considering the lack of information allowing a correct discrimination. While Atty et al. [AHL*06] somewhat alleviate the problem by maximum combining the soft shadow maps for the nearest front facing and farthest back facing occluder faces, only Guennebaud et al. [GBP06] explicitly address the problem algorithmically. For each micropatch they look at the left and bottom neighbors in the depth map and extend the micropatch appropriately to these neighbors' borders. In Section 4, we introduce an alternative approach which implicitly closes any gaps and further improves on the reconstruction of occluders from the depth map.

Fourth, to ensure interactive performance, the number of micropatches that are backprojected for each pixel to be shaded has to be limited. However, in some cases this upper bound is necessarily too small and, as with all sampling-based approaches, artifacts can appear. To alleviate this problem, Guennebaud et al. [GBP06] build a hierarchical shadow map (HSM) via a mipmap-like reduction storing the minimum and maximum depth at each level. They then use this HSM to derive a tighter bound of the depth map region which contains all potentially occluding micropatches, thus reducing the number of necessary backprojections. To enforce an upper bound of considered micropatches, a coarser level of the HSM can be sampled instead of the original depth map. Furthermore, the HSM is used to detect that a point is completely lit or within the umbra, thus limiting the number of pixels for which micropatch backprojection has actually to be performed. We suggest another multi-scale representation, termed multi-scale shadow map, which often results in much better classification results and hence drastically reduces the number of pixels subjected to actual soft shadow computations.

3. Bitmask soft shadows

Our new algorithm, which we term bitmask soft shadows (BMSS), computes dynamic soft shadows with correct occluder fusion cast by rectangular area light sources in real-time. We first render a single shadow map from a sample point on (or behind) the light source which is usually placed at its center. Similar to the approach of Guennebaud et al. [GBP06], we interpret texels of the obtained depth map as micropatches and backproject them onto the light source to determine the caused degree of occlusion.

However, as discussed in the previous section, simply accumulating the occluded light areas ignores potential overlaps, which can lead to severe artifacts like those in Fig. 2(b) (see also Fig. 5). We therefore take a different approach often encountered in ray tracing and sample the light source visibility via several point-to-point relations. More precisely, we place sample points on the light source and use a bit field to track which of them is occluded. The resulting occlusion bitmask provides a discrete representation of the light source's

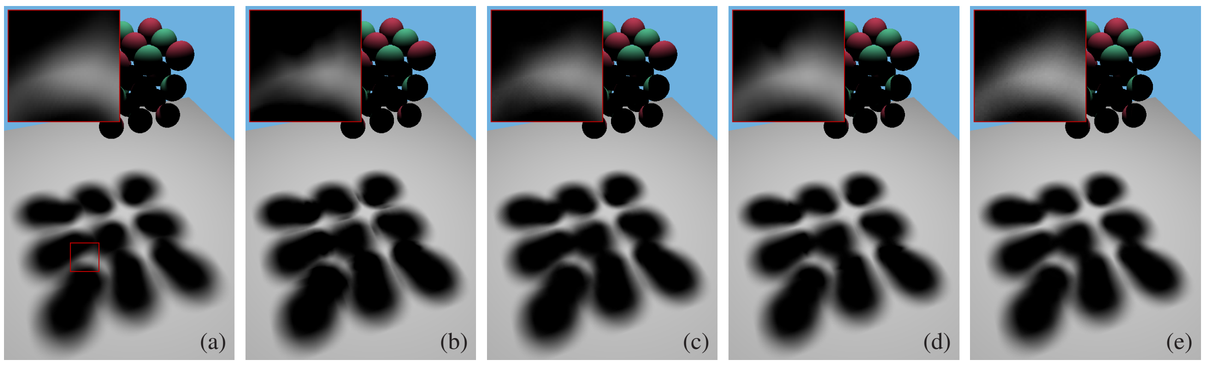

Figure 2: *Array of $3 \times 3 \times 3$ balls: (a) reference image; (b) accumulating occlusion of backprojected micropatches [GBP06] (c) BMSS with micropatches (16×16, jittered); (d) accumulating occlusion of backprojected microquads (no BMSS); (e) BMSS with microquads (16×16, jittered).*

occlusion and we use the number of set bits to determine the light's visibility.

Thanks to the new integer processing capabilities of DirectX 10 class graphics hardware [Bly06], bit operations became practical within shaders. Nevertheless, to allow for fast updates of the occlusion bitmask, we cannot arbitrarily place samples on the light and we also have to limit the number of samples. As illustrated in the bottom row of Fig. 3, we therefore considered regularly placed, uniformly spaced sampling points for bit fields of size 8×8, 16×16 and 32×32.

Since micropatches are axis-aligned rectangles, the bitmask can be updated efficiently to incorporate the occlusion due to a new patch. We first map the micropatch's axial extent into bit ranges via scale, bias and integer conversion operations and then derive a bitmask for the patch which gets OR-ed with the occlusion bitmask. Note that in case of overlapping micropatches, the same bit gets set multiple times, i.e. occluder fusion is automatically dealt with correctly.

While 16×16 sample points often offer enough levels of visibility discrimination, a strictly regular sampling pattern is usually suboptimal and can lead to clearly visible discretization artifacts. We alleviate this by jittering the sample positions, but for performance reasons, we settled for a regular jitter pattern instead of a truly random one. Note that the resulting sampling pattern is identical to an 8×8 tiling of the 2×2 rotated grid super sampling (RGSS) pattern, which is known to offer good anti-aliasing quality for nearly vertical and horizontal edges [AMH02]. Similarly, our jittered sampling pattern is well-suited to deal with the encountered axis-aligned rectangles. In particular, it often closely matches the quality of the regular 32×32 sampling pattern.

3.1. Post filtering of the visibility buffer

Since we approached the visibility determination as a point sampling problem, a hard transition occurs when a sample finally gets hit and hence a bit becomes set. This might give rise to discretization artifacts, especially if more than one bit changes state when transitioning between two points corresponding to two adjacent pixels in the final image. Note that such a case is not that unlikely because micropatches are axis-aligned and the employed sampling pattern is (semi-)regular. Hence artifacts are particularly severe if a large edge of a shadow caster is in good alignment with a border of the light source as in the scene depicted in Fig. 9. In practice, however, discretization artifacts are often acceptable, especially since they are usually masked by the texture of the shadow receiving surface [FPSG97] and hence remain (largely) imperceptible.

To alleviate such artifacts nevertheless, we exploit the observation that while simply accumulating occluded areas is often wrong, the resulting (unclamped) final occlusion estimate changes rather smoothly. More precisely, we write the visibility values obtained via the occlusion bitmask (B) and via the accumulated area (A) into a visibility buffer. We then use the area-based values A to drive a smoothing of the visibility values B derived from the bitmask.

In a single filter pass, for each pixel first the central differences in x-direction are derived for both A and B. If their signs match and the pixel's area-based value A_0 is between those of the two neighboring pixels (A_{-1}, A_1), its relative position is imposed on the pixel's bitmask-based value B_0:

$$B_0^x = \frac{A_0 - A_{-1}}{A_1 - A_{-1}} (B_1 - B_{-1}) + B_{-1}.$$

The same is done in y-direction and finally that value among B_0^x and B_0^y is adopted as new B_0 value which constitutes the smaller actual change.

As illustrated by the results in Fig. 3, post filtering often helps reducing discretization artifacts significantly. While we didn't observe this heuristic approach introducing any new objectionable artifacts with our test scenes, we reckon

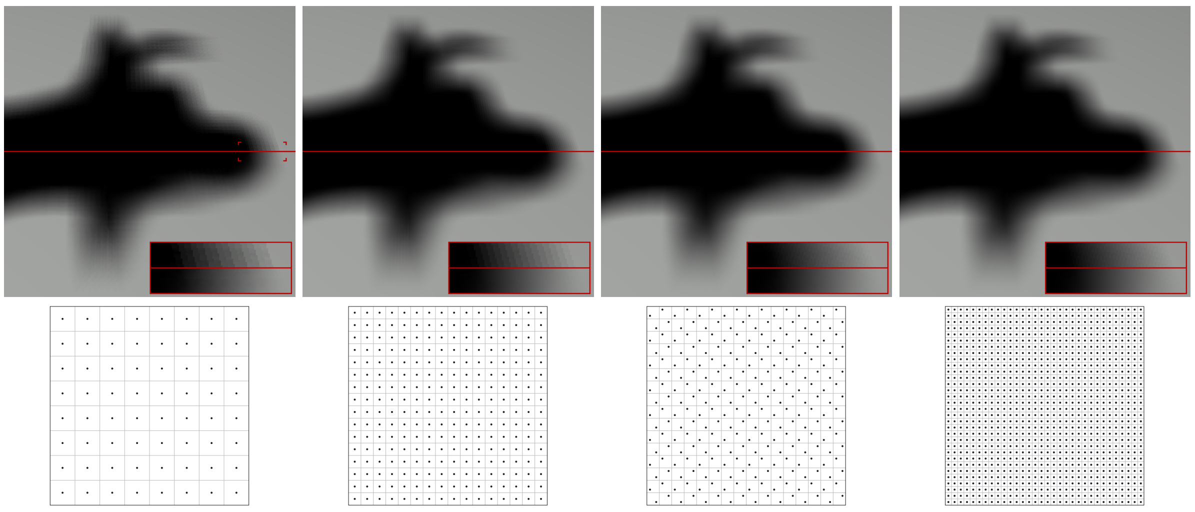

Figure 3: *Top row: soft shadow of a cow model's head without (upper half) and with (lower half) visibility buffer filtering (5 filter passes). Bottom row: placement of used sampling points.*

that more advanced filtering schemes might be necessary for more intricate scenes but leave this as future work.

3.2. Multiple depth maps

As presented so far, BMSS only used information from a single depth map. Concerning visual quality, our algorithm hence improves merely on artifacts due to overlaps encountered in related approaches. We acknowledge that these artifacts might be acceptable in many cases though, because they can be rarely noticeable. This is mainly because the possibility of overlap is kept small compared to techniques like soft shadow volumes in the first place by only processing occluders visible from a dedicated point on the light source.

On the other hand, since our algorithm correctly handles occluder fusion, a new class of applications becomes possible. In particular, we can incorporate occluder information extracted from multiple depth maps in the visibility determination process. This enables us to basically capture all occluders and correctly account for them. Hence BMSS lift alternative algorithms' severe limitation of only processing those occluders visible from a single sample point. As demonstrated by the well-known single silhouette artifact setup [AAM03] in Fig. 9, this restriction can lead to objectionable artifacts that deprive the image of important depth cues.

To capture additional occluders, one could render shadow maps from multiple sample points on the light source. An easier way is to perform depth peeling from a single sample point. Note that to keep the cost of additional render passes low, a smaller frustum can be used for all but the first depth layer. For instance, one might focus only on objects directly below the light source since usually most occlusion artifacts occur there.

It is now also possible to use a separate depth map per object or group of object. For instance, in case of a scene composed of a moving object and other objects currently not moving, we can derive occlusion bitmasks for them at different rates and combine these bitmasks each frame to determine the correct light visibility.

3.3. Multi-colored light sources

BMSS can readily be adapted to deal with multi-colored light sources. In case of few different colors, we use bitmasks identifying sample points of the same color, perform a bitwise AND with the occlusion bitmask and count the set bits. Fig. 4 shows an example using a bit field of size 32×32. Note that this extension introduces far less overhead than approaches using a 4D texture or a summed area table [GBP06] which necessitates multiple texture access per backprojected micropatch.

In case of arbitrarily textured light sources with many different colors, it becomes more efficient to perform a texture look-up for each group of 8 bits to derive the color for the visible fraction of the light source.

4. Backprojecting microquads

Reconstructing occluders' geometry via micropatches suffers from several problems because the patches are chosen to be parallel to the light area plane such that their backprojections are axis-aligned rectangles. As a consequence, gaps are introduced which require special processing, and occluders are often overestimated, leading to noticeable overestimations of the penumbra's extent (cf. Fig. 2).

To alleviate these shortcomings, we propose a new interpretation of the shadow map data. Instead of considering each texel of the depth map as a micropatch, we consider the texels' unprojected centers as the vertices of *microquads*. Such a quad gets only backprojected if all four vertices are closer to the light source than the point for which light visibility is determined.

We note that this alternative interpretation in some respect comes down to replacing nearest neighbor interpolation with bilinear interpolation. It is hence not surprising that microquads adapt better to the actual geometry than micropatches (cf. Fig. 5). In particular, since adjacent quads share a common boundary no unwanted gaps occur in the first place.

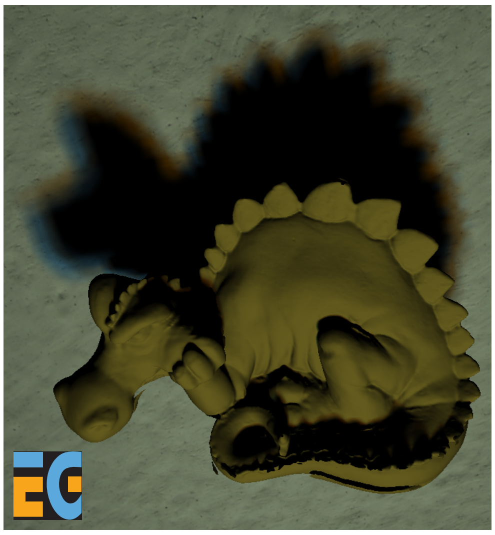
Figure 4: *Phlegmatic dragon illuminated by an EG logo light source (inset).*

Moreover, using microquads is less prone to surface acne. Imagine a plane which is visible from the light's center and not perpendicular to the z-axis of the depth map's frustum. In case of using a micropatch interpretation, many points on that plane which don't coincide with a depth map sample point get partially occluded by the micropatch corresponding to the nearest sample point closer to the light, thus suffering from incorrect self-shadowing. This is especially hard to avoid via biasing (without causing artifacts in other parts of the image) if patches from coarser hierarchy levels get considered. Microquads, on the other hand, don't cause any artifacts in this setup.

While determining the area of an arbitrary 2D quad is almost as simple as in case of a rectangle, accurately clipping the backprojection of a microquad to the light area is rather expensive. We hence resort to the simple approach of

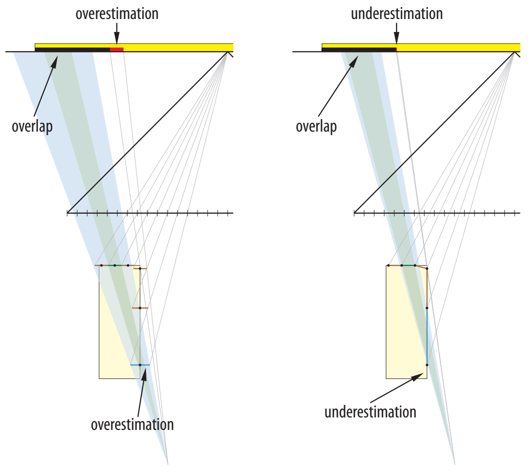

Figure 5: *While using micropatches (left) often suffers from overestimating the occluders, microquads (right) can lead to some underestimation but usually adapt better to the sampled geometry. Note that with both interpretations overlaps can occur as demonstrated by the blue and green primitives.*

just clamping the projected vertex coordinates (see Fig. 6). While this approximation can introduce minor errors in the covered area, these imprecisions only occur at the boundary of a mesh of connected microquads. In particular, no gaps or overlaps occur due to this clamping step since the vertices are adapted consistently across microquads.

In case of BMSS, however, we approximate each microquad by fitting an axis-aligned rectangle to the center points of the quad's edges since updates to the occlusion bitmask for arbitrary quads are quite involved. While this alleviates area overestimation compared to using the microquad's bounding box, it breaks the connectedness of diagonal neighbors, thus introducing minor gaps and overlaps.

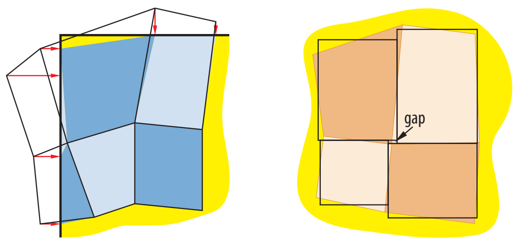

Figure 6: *Left: Backprojected microquads are clipped to the light area by projecting their vertices onto the nearest boundary edge via clamping. Note that the microquad mesh remains watertight and that no overlaps are introduced. Right: To allow for fast bitmask updates, we approximate each microquad with a rectangle defined by the centers of the quad's edges.*

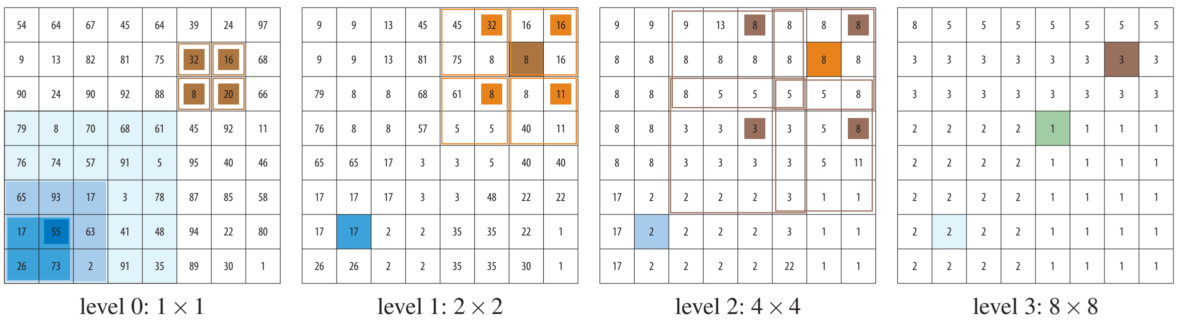

Figure 7: *Multi-scale shadow map (z_{min} channel) of resolution 8×8: The blue texels in levels 1–3 contain the minimum values of the identically colored regions in level 0. Each fully colored brownish texel is derived from the values of the texels in the previous level of the same shade which together cover a region as indicated by the colored rectangles. The overall minimum can be accessed via the green texel.*

Note that the same problem occurs when filling gaps via micropatch extension.

Despite these simplifications, dealing with microquads is slightly more expensive than operating with micropatches. This is largely due to more complex computations and not because of an often marginally increased number of required texture fetches. As when using micropatches and performing gap filling, for each micro-primitive usually only two new texels have to be accessed since the remaining ones are known from the previously processed micro-primitive.

5. Improved search area refinement

Since both updating a bitmask and using microquads incurs some additional cost, it becomes very important to keep the number of those backprojected micro-primitives to a minimum that don't intersect the light area at all and hence don't contribute to the final occlusion value. While the HSM helps reducing the search area of considered potential micro-primitives, the resulting bounds are usually quite conservative.

We therefore suggest an alternative multi-scale representation of the depth map which is not pyramidal like HSM and classical mipmaps, but retains the original resolution across all levels. In this multi-scale shadow map (MSSM), a texel at level i stores the minimum and maximum depth values in a neighborhood region of size $2^i \times 2^i$ centered around the texel. As illustrated in Fig. 7, each texel at level $i > 0$ can be derived from four texels t_{uv} of level $i-1$. Because the neighborhood region is clamped to the depth map extent, a clamp-to-edge texture wrapping mode is employed when accessing texels t_{uv}. Resulting overlaps of neighborhood regions don't cause any problems since only their extrema are combined.

We note that the MSSM can be considered as a variant of the N-buffer [Déc05]. The main difference is that we store a neighborhood region's depth range at its (discretized) center instead of its lower left corner. Consequently, the MSSM provides more information than an N-buffer since no redundant data is stored. In an N-buffer, the 2^i top-most rows and right-most columns in the i-th level are identical to the corresponding elements in all higher levels since the region information is gathered about doesn't change any longer for these elements. Ultimately, with our MSSM, we get better results for sample points where the (unclamped) neighborhood region crosses the depth map's border.

Like Guennebaud et al. [GBP06], we initialize the search area to the (clamped) projection of the light source onto the depth map's near plane. A first bound $[z_{\min}, z_{\max}]$ for the search area's depth range is then obtained via a single MSSM sample. If the currently considered point $\mathbf{p}$ is outside this range, it is either completely lit or in umbra, rendering any micro-primitive processing unnecessary. Otherwise, $z_{\min}$ is used to refine the search area. We then use four samples from the appropriate MSSM level such that their (partially overlapping) neighborhood regions tightly cover the search area to get a more accurate depth range bound. This is finally used to further refine the search area before starting with any micro-primitive backprojection.

When constructing micro-primitives from coarser levels $i > 0$ (for each fragment only a single level i is used), the actual sampling positions are restricted to the same grid implicitly imposed by the HSM. Otherwise, neighboring pixels in the final image would often sample the depth map at an identical scale but at slightly translated and hence different sample points, which leads to artifacts.

As demonstrated in Fig. 8, the MSSM can significantly reduce the search area size compared to the HSM—often even to zero such that no micro-primitive processing is required at all. On the down side, the MSSM consumes more memory than the HSM since the resolution is not reduced across levels. Depending on the application context one might hence prefer to adopt a hybrid approach where a pyramidal reduction is performed for the finer levels before switching to a MSSM-like representation for the remaining levels.

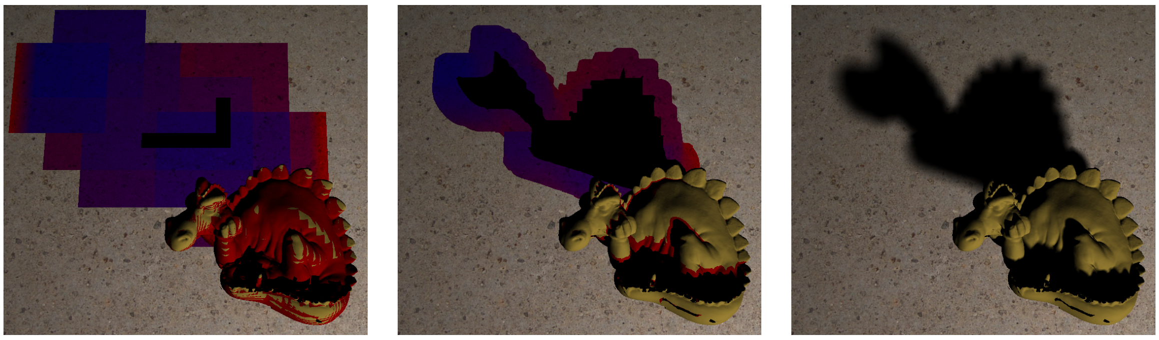

Figure 8: *HSM (left) vs. MSSM (center): pixels for which some micropatches have to be backprojected to determine the actual degree of the light's occlusion are highlighted (red: at most 6×6 patches at an effective shadow map resolution of 256^2 need to be processed, blue: at least 20×20 patches). Right: resulting soft shadow.*

6. Implementation details

To be able to utilize the required DirectX 10 class features of the employed NVIDIA GeForce 8800 GTS, we resorted to implement the described algorithm with OpenGL using GLSL. The MSSM gets stored in a 2D array texture because this enables the dynamic selection of the sampled level within a shader.

Regarding the bit field operations, we tried to come up with well-optimized versions. Nevertheless, updating the bitmask for the jittered 16×16 sampling pattern still requires 131 scalar instructions. In case of tracking visibility by a 32×32 bit field, 32 scalar registers are required just for storing the bit field. This high register count seems to negatively affect the number of concurrently processed threads, thus limiting overall speed. Indeed, when performing the same instructions but using only 16 scalar registers, the performance increases by roughly 75%. We ended up with employing temporary array variables which are stored in slower local memory but enable a higher degree of parallelism, achieving a speed-up of more than 50% compared to the register-based variant.

7. Results

In the following we present some results obtained with an NVIDIA GeForce 8800 GTS. All timings are reported for a viewport of size 1024×768. Since our algorithm aims at real-time applications, we capped the number of considered micro-primitives to retain at least interactive frame rates. In some cases we hence sample a coarser HSM or MSSM level to construct less but larger micro-primitives.

Fig. 2 demonstrates that while micropatches often give rise to an overestimation of the soft shadow extent, microquads usually yield results much closer to the reference. Noticeable overlapping artifacts, however, can only be avoided with correct occluder fusion handling as provided by BMSS.

In the simple but quite demanding scene depicted in Fig. 9, employing depth peeling to capture additional occluders allows us to come quite close to the reference obtained from averaging 1024 hard shadows. It also becomes obvious that determining the occlusion due to the closest front faces and due to the farthest back faces separately and taking their maximum does not necessarily yield correct results. Table 1 lists the achieved frame rates when using the HSM to refine the search area. In case of BMSS, post filtering with five filter passes was performed, where a single pass takes roughly 0.35 ms. Note that when the number of depth layers doubles, the frame rate almost halves because the number of processed micropatches roughly doubles, too.

Standard backprojection [GBP06]	39.7
BMSS, 8×8	37.2
BMSS, 16×16	23.5
BMSS, 16×16 jittered	19.2
BMSS, 32×32	9.6
BMSS, 16×16 jittered, two depth layers	11.1

Table 1: *Measured frame rates in Hz for the single silhouette artifact setup in Fig. 9. Each depth map was of size 512^2; the number of considered micropatches was restricted to 16×16 within each depth map.*

Table 3 reports frame rates with post filtering disabled for three different scenes. The cow scene of Fig. 10 is representative of setups where a rather small area light is employed and the number of considered micro-patches is restricted to a degree which ensures high frame rates. Since thus often coarser levels of the HSM or MSSM are employed for micro-primitive construction anyway, the depth map resolution was chosen comparatively small in the first place.

The measured rendering times for the four presented occlusion bitmask sample patterns along with the attained visual quality (see Fig. 3) suggest that using a bit field of size

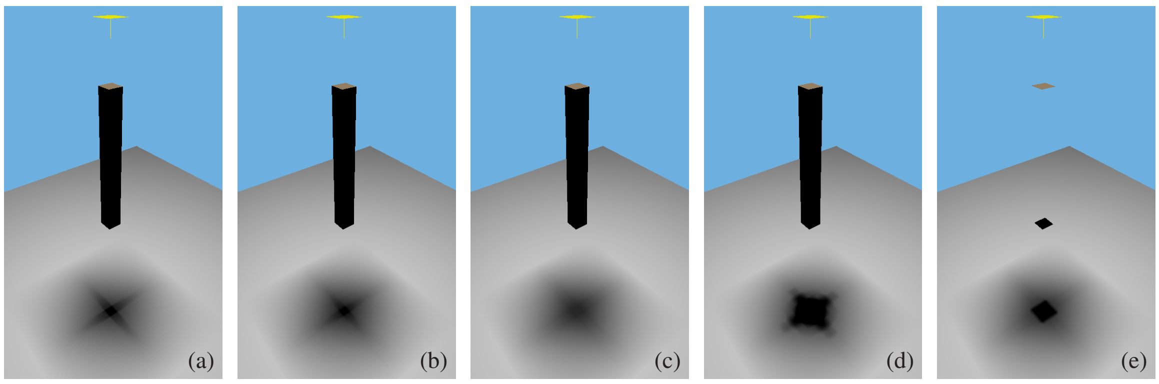

Figure 9: *Single silhouette artifact setup: (a) reference image; (b) BMSS with two depth layers (16 × 16, jittered); (c) standard backprojection algorithm [GBP06]; (d) maximum of the occlusion accumulated separately for two depth layers as suggested by Atty et al. [AHL*06]; (e) maximum of the occlusion due to two separately considered planar light blockers.*

16 × 16 with jittered sample points provides the best trade-off between quality and speed.

Figure 10: *Cow model and its soft shadow.*

If the scene content allows for a decisive reduction of the search area's size when employing a MSSM instead of a HSM, the larger construction time for the MSSM (cf. Table 2) gets not only completely amortized but the smaller search area also directly translates into a higher frame rate.

Size	256^2	512^2	1024^2	2048^2
HSM	0.92	1.06	1.19	3.61
MSSM	0.32	1.22	5.44	24.0

Table 2: *Times in ms required for constructing the remaining HSM and MSSM levels for a given depth map.*

As already detailed in Section 4, dealing with microquads turns out to be slightly slower than utilizing a micropatch interpretation. Similarly, the improved visual quality offered by BMSS comes at a certain cost since bitmask updates for larger bit fields are more expensive than determining an area (which is additionally done in case post filtering is enabled). However, for many scenes, thanks to the speed-ups gained by using the MSSM, with our jittered 16 × 16 variant, we nevertheless remain roughly on par with ordinary HSM-guided micropatch occlusion accumulation concerning rendering speed while providing an improved visual quality.

8. Conclusion and future work

We have presented a new algorithm for real-time soft shadows based on tracking light visibility with a bit field. Providing a solution to the important occluder fusion problem, we improve on the visual quality attainable with current state-of-the-art soft shadowing techniques. We note that, in principle, our algorithm is not restricted to working with micro-primitives. For instance, it would be interesting to apply our technique to soft shadow volumes.

We have also suggested two improvements whose applicability is not restricted to our BMSS but which can readily be applied to related algorithms. First, with microquads obtained from an alternative interpretation of shadow map data, occluder geometry is often better reconstructed, resulting in improved visual quality. Second, effective narrowing of the search area via the MSSM representation can substantially increase the frame rate or enable a higher soft shadow quality at an unchanged speed, respectively.

In future work, we would like to investigate whether and how the currently required computational effort can be reduced significantly while retaining or even improving on the currently achieved quality.

Acknowledgements

This work was funded by the European Union within the CROSSMOD project (EU IST-014891-2).

References

[AAM03] Assarsson U., Akenine-Möller T.: A geometry-based soft shadow volume algorithm using

Scene	Fig. 2		Fig. 8		Fig. 10	
Depth map size	1024^2		1024^2		512^2	
Max. number of considered micro-primitives	18×18		20×20		12×12	
Search area refinement via	HSM	MSSM	HSM	MSSM	HSM	MSSM
Micropatches (area accumulation)	19.8	21.0	13.3	20.3	42.7	80.1
Microquads (area accumulation)	19.6	20.8	12.7	18.6	37.3	71.4
BMSS, 8×8, micropatches	20.2	18.3	13.9	18.1	45.8	71.0
BMSS, 16×16, micropatches	12.7	10.3	9.5	12.1	33.9	44.4
BMSS, 16×16 jittered, micropatches	10.0	10.1	7.5	11.8	26.6	43.4
BMSS, 32×32, micropatches	5.3	5.9	4.3	6.7	15.9	25.7
BMSS, 16×16 jittered, microquads	8.1	8.1	5.9	9.3	20.4	34.6

Table 3: *Frame rates in Hz obtained with our implementation. Note that both the extent of the umbra and the penumbra's size vary significantly between the scenes.*

graphics hardware. *ACM Transactions on Graphics 22*, 3 (2003), 511–520.

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

[AHT04] ARVO J., HIRVIKORPI M., TYYSTJÄRVI J.: Approximate soft shadows with an image-space flood-fill algorithm. *Computer Graphics Forum 23*, 3 (2004), 271–280.

[AMH02] AKENINE-MÖLLER T., HAINES E.: *Real-Time Rendering*, 2nd ed. A.K. Peters Ltd., 2002.

[ASK06] ASZÓDI B., SZIRMAY-KALOS L.: Real-time soft shadows with shadow accumulation. In *Eurographics 2006 Short Presentations* (2006), pp. 53–56.

[Bly06] BLYTHE D.: The Direct3D 10 system. *ACM Transactions on Graphics 25*, 3 (2006), 724–734.

[BS02] BRABEC S., SEIDEL H.-P.: Single sample soft shadows using depth maps. In *Proceedings of Graphics Interface 2002* (2002), pp. 219–228.

[BS06] BAVOIL L., SILVA C. T.: Real-time soft shadows with cone culling. In *ACM SIGGRAPH 2006 Sketches and Applications* (2006).

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

[Déc05] DÉCORET X.: N-buffers for efficient depth map query. *Computer Graphics Forum 24*, 3 (2005), 393–400.

[FBP06] FOREST V., BARTHE L., PAULIN M.: Realistic soft shadows by penumbra-wedges blending. In *Proceedings of Graphics Hardware 2006* (2006), pp. 39–47.

[Fer05] FERNANDO R.: Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches and Applications* (2005).

[FPSG97] FERWERDA J. A., PATTANAIK S. N., SHIRLEY P., GREENBERG D. P.: A model of visual masking for computer graphics. In *Proceedings of ACM SIGGRAPH 97* (1997), pp. 143–152.

[GBP06] GUENNEBAUD G., BARTHE L., PAULIN M.: Real-time soft shadow mapping by backprojection. In *Proceedings of Eurographics Symposium on Rendering 2006* (2006), pp. 227–234.

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

[KD03] KIRSCH F., DÖLLNER J.: Real-time soft shadows using a single light sample. *Journal of WSCG 11*, 2 (2003), 255–262.

[LA05] LAINE S., AILA T.: Hierarchical penumbra casting. *Computer Graphics Forum 24*, 3 (2005), 313–322.

[LB06] LOOP C., BLINN J.: Real-time GPU rendering of piecewise algebraic surfaces. *ACM Transactions on Graphics 25*, 3 (2006), 664–670.

[OP05] OLIVEIRA M. M., POLICARPO F.: *An Efficient Representation for Surface Details*. Tech. Rep. RP-351, Universidade Federal do Rio Grande do Sul, 2005.

[PSS98] PARKER S., SHIRLEY P., SMITS B.: *Single Sample Soft Shadows*. Tech. Rep. UUCS-98-019, University of Utah, 1998.

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)* (1987), vol. 21, pp. 283–291.

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