Efficient Screen Space Anisotropic Blurred Soft Shadows

Zhongxiang ZHENG†, Nonmember and Suguru SAITO†∗, Member

SUMMARY Shadow mapping is an efficient method to generate shadows in real-time computer graphics and has broad variations from hard to soft shadow synthesis. Soft shadowing based on shadow mapping is a blurring technique on a shadow map or on screen space. Blurring on screen space has an advantage for efficient sampling on a shadow map, since the blurred target array has exactly the same coordinates as the screen. However, a previous blurring method on screen space has a drawback: the generated shadow is not correct when a view direction has a large angle to the normal of the shadowed plane. In this paper, we introduce a new screen space-based method for soft shadowing that is fast and generates soft shadows more accurately than the previous screen space soft shadow mapping method. The resultant images show shadows produced by our method just stand in the same place, while shadows by the previous method change in terms of penumbra while the view moves. Surprisingly, although our method is more complex than the previous method, the measurement results of the calculation time show our method is almost the same performance. This is because it controls the blurring area more accurately and thus successfully reduces multiplications for blurring.

key words: computer graphics, shadow mapping, soft shadow, screen space, real time rendering

1. Introduction

Generating shadows is one of the most important factors for rendering realistic scenes in computer graphics. While shadows cast by a point light, whose edges are sharp, are called hard shadows, shadows cast by an area light, whose edges are soft, are called soft shadows. In real life, we see soft shadows because almost no light sources in the real world are point light sources. Therefore, generating soft shadows is an important factor for rendering realistic scenes in computer graphics.

Shadow mapping is an efficient method to generate shadows in real-time computer graphics. The original shadow mapping is for hard shadow rendering, but there are extended versions for soft shadow synthesis. These extensions generate soft shadows from hard shadows by blurring on a shadow map or on screen space. The method for blurring on screen space [1] has faster calculation time than that on a shadow map. However, this previous screen space-based method has a drawback: the generated shadow is not correct when a view direction has a large angle to the normal of the shadowed plane.

In this paper, we introduce a new screen space-based method to generate soft shadows [2]. This method generates soft shadows more correctly than the previous screen space-based soft shadowing method with controlling blurring kernels by considering the normals of shadow receivers and the view direction that changed with each screen pixels by a perspective projection. This method is efficient for real time dynamic scene rendering because it does not use any pre-filtering technique. As the experiment section shows, its calculation performance is surprisingly almost the same as that of the previous method, though it is more complex.

2. Related Work

Shadow volume [3] and shadow mapping [4] are major methods for rendering hard shadows in real time. Shadow volume is a geometry-based method. While aliasing does not appear on shadow edges, its computational cost is more affected by the complexity of geometry than image-based methods. However, this geometry-based method is better at generating accurate soft shadows. There are geometry-based methods for soft shadows that create the geometry for penumbra regions and compute smooth shadows on receivers [5]. Generally, although their computational costs are more expensive, geometry-based methods have better shadow quality and accuracy than image-based methods. Several improved methods [6], [7] have also been proposed. However, their computational cost is sensitive to the complexity of the geometry.

Shadow mapping [4], an image-based method, is one of the most popular methods for rendering hard shadows in real time. This is an image-based method. The increase of the computational cost of this method is lower than the increase of the cost of the geometry-based methods when the scene geometry becomes complex. However, it often suffers from the aliasing of shadow.

Percentage-closer filter (PCF) [8] and Variance Shadow Maps (VSM) [9] solve the aliasing problem by softening edges of shadows with filtering techniques.

On the basis of PCF or VSM, several methods have been proposed for rendering soft shadows in real time. One image-based method for generating perceptually accurate soft shadows is percentage-closer soft shadows (PCSS) [10]. PCSS consists of three steps. First is searching for occluders of receiver points from a shadow map. Second is estimating the penumbra size at each receiver point from distance among a light source, occluder, and receiver. Third is performing typical PCF on the shadow map using an adaptive
kernel size in accordance with the penumbra size. In steps 1 and 3, sampling is performed on the shadow map. The number of samplings increases in accordance with the light size and penumbra size. This increase in sampling causes an increase in rendering time. Summed area variance shadow maps (SAVSM) [11] can generate arbitrary penumbra size in constant time by using summed-area table (SAT) and VSM. However, the computational cost to set up SAT is affected by the resolution of the shadow map. Therefore, in the case of scenes in which lights and objects move, it is not appropriate because SAT has to be updated every rendering time. Variance Soft Shadow Mapping (VSSM) [12] accelerates the occluder search process with its formula based on VSM while SAVSM perform occluder search by brute-force sampling of the shadow map. However, VSSM also uses SAT, thus it has the same drawback as SAVSM.

Recently, Exponential Soft Shadow Mapping has been introduced [13]. This solves an overflow problem of SAT, when applying a large scene, by tiling SAT. However, it is also inappropriate for dynamic scenes because of the use of pre-filtering.

Annen et al. [14] also utilized a local averaging filter on a shadow map, which is applied to Convolution Shadow Maps (CSM) that store approximations of shadow test functions [15]. This algorithm achieves high quality soft shadows, but it also needs to store approximations calculated by pre-filtering.

Back-projection methods [16], [17] are also image based methods. They are more physically accurate than PCSS based methods due to their accurate approximation. These methods reconstruct occluders in world space from a shadow map obtained with a camera positioned in the center of the light source. However, these methods are computationally more expensive than PCSS based methods.

Recently, Shen et al. introduced a filtering method on a shadow map for high quality soft shadows. To realize high quality soft shadows, it employs virtual shadow maps and anisotropic filters that take into account of a projected pixel shape on the shadow map [18].

On the other hand, there are filtering methods that work on screen space and allow the use of a large shadow map, without dramatically affecting the frame rate. This group includes Image-Space Gathering (ISG) [19] and Screen-Space Percentage-Closer Soft Shadows (SSPCSS) [1]. These methods perform analogous algorithms of PCSS on screen space. SSPCSS estimates penumbra sizes on screen space and also blurs hard shadows on screen space with an adaptive bilateral Gaussian filter in accordance with penumbra sizes. The advantage of SSPCSS is that it uses a separable bilateral Gaussian filter [20], [21] as an approximation of the bilateral Gaussian filter. This approximation is highly efficient but less accurate.

When a viewpoint moves, there filtering methods that work on screen space and allow the use of a large shadow map, without dramatically affecting the frame rate. This group includes Image-Space Gathering (ISG) [19] and Screen-Space Percentage-Closer Soft Shadows (SSPCSS) [1]. These methods perform analogous algorithms of PCSS on screen space. SSPCSS estimates penumbra sizes on screen space and also blurs hard shadows on screen space with an adaptive bilateral Gaussian filter in accordance with penumbra sizes. The advantage of SSPCSS is that it uses a separable bilateral Gaussian filter [20], [21] as an approximation of the bilateral Gaussian filter. This approximation is highly efficient but less accurate.

When a viewpoint moves, there are filtering methods that work on screen space and allow the use of a large shadow map, without dramatically affecting the frame rate. This group includes Image-Space Gathering (ISG) [19] and Screen-Space Percentage-Closer Soft Shadows (SSPCSS) [1]. These methods perform analogous algorithms of PCSS on screen space. SSPCSS estimates penumbra sizes on screen space and also blurs hard shadows on screen space with an adaptive bilateral Gaussian filter in accordance with penumbra sizes. The advantage of SSPCSS is that it uses a separable bilateral Gaussian filter [20], [21] as an approximation of the bilateral Gaussian filter. This approximation is highly efficient but less accurate.

When a viewpoint moves, there are filtering methods that work on screen space and allow the use of a large shadow map, without dramatically affecting the frame rate. This group includes Image-Space Gathering (ISG) [19] and Screen-Space Percentage-Closer Soft Shadows (SSPCSS) [1]. These methods perform analogous algorithms of PCSS on screen space. SSPCSS estimates penumbra sizes on screen space and also blurs hard shadows on screen space with an adaptive bilateral Gaussian filter in accordance with penumbra sizes. The advantage of SSPCSS is that it uses a separable bilateral Gaussian filter [20], [21] as an approximation of the bilateral Gaussian filter. This approximation is highly efficient but less accurate.
The temporal penumbra size on screen space, \( w_{ps} \), at a pixel is calculated from \( w_p \) of the corresponding point in the world space as

\[
\begin{align*}
\frac{w_p}{d_{ps}} &= \frac{d_{ps}}{w_l}. \\
\frac{w_{ps}}{z_v} &= \frac{w_p d_{screen}}{z_v}, \text{ and} \\
\frac{d_{screen}}{2} &= \frac{1}{2 \tan \left( \frac{fov}{2} \right)}.
\end{align*}
\]

where \( z_v \) is the depth from the viewpoint at the focusing pixel and \( \text{fov} \) is the view angle of a camera.

The size \( w_{ps} \) is used as the diameter of the variable-size filter to blur the projected shadow map on screen by SSPCSS. However, the angle between the normal of the shadowed surface and the view direction is not taken into consideration for it. As a disk becomes an ellipse when looked at from an angle (Fig. 2), we utilize an anisotropic filter that contours an ellipse whose minor and major radii are \( r_{lg} \) and \( r_{sh} \). In our previous method [2], [23] \( r_{sh} \) and \( r_{lg} \) are defined as

\[
r_{sh} = (N_v \cdot N_s)w_{ps}, \quad \text{and} \quad r_{lg} = \frac{w_{ps}}{z_v}.
\]
Then the shorter and the longer radii in $r_{sh}$ and $r_{lg}$ are equivalent, the unknown parameters are determined, whose general expression is Eq. (10), are obtained.

Assuming that the point on the shadowed plane, its normal, and the penumbra size for a focusing point $p_0$ on screen space are $p$, $n$, and $w_p$, respectively, the contour disk that is corresponding to the blurring kernel at $p_0$ is the intersection between the plane $P$ and the sphere $S$ as Eqs. (7) and (8).

$$P : n_x(x - p_x) + n_y(y - p_y) + n_z(z - p_z) = 0$$
$$S : (x - p_x)^2 + (y - p_y)^2 + (z - p_z)^2 - w_p^2 = 0$$

The disk is also the intersection between $P$ and the elliptic cone $E$, whose apex is the camera position (the origin), as

$$E : \left(\frac{(x - p'_{0x}, z) \cos \theta + (y - p'_{0y}, z) \sin \theta}{r_a}^2\right) + \left(\frac{-(x - p'_{0x}, z) \sin \theta + (y - p'_{0y}, z) \cos \theta}{r_b}^2\right) - z^2 = 0,$$

where $p'_{0x}$, $\theta$, $r_a$, $r_b$ are unknown parameters. Substituting the left-hand side of Eq. (7) to $z$ of Eqs. (8) and (9), two equations, whose general expression is Eq. (10), are obtained. Since these are equivalent, the unknown parameters are derived.

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$

Then the shorter and the longer radii in $r_a$ and $r_b$ and those corresponding directions are assigned to $r_{sh}$, $r_{lg}$, $A_{sh}$, and $A_{lg}$ respectively. Note that the derived centroid $p'_0$ of the ellipse is slightly different from $p_0$, but we approximate $p'_0 \approx p_0$ and make the kernel centroid $p_0$.

### 3.2 Separable Bilateral Anisotropic Gaussian Blur

If a 2D Gaussian filter is applied to blur hard shadows on screen space without considering occluder edges, then light bleeding appears, as shown in Fig. 4 (a). To solve this problem, SSPCSS adopts a local adaptive separable bilateral Gaussian filter, which is an approximated version of a local adaptive bilateral filter, as an edge-aware filter. The weight of the local adaptive bilateral filter depends on two values. The first is the depth difference between the depth from a camera of a pixel and that of the center pixel in the filter. The second is the distance between the two pixels on screen space. The brightness of shadow $S_{dw}$ at $p_0$ on screen space is calculated by the adaptive bilateral filter expressed as

$$S_{dw}(p_0) = \frac{\sum_{p \in P(p_0)} g_f(p - p_0) g_s(d_p(p) - d_p(p_0)) S_{sh}(p)}{\sum_{p \in P(p_0)} g_f(p - p_0) g_s(d_p(p) - d_p(p_0))},$$

where $P(p_0)$ is a set of samples in a local area whose center is $p_0$ on screen space, $d_p(\cdot)$ is the depth from the camera, $g_f$ is a two dimensional Gaussian function whose argument is a vector on screen space, $g_s$ is a Gaussian function whose argument is the difference between depths from a viewpoint, and $S_{sh}(\cdot)$ is a hard shadow function. The standard deviation of $g_f$ is decided by $w_p$ at $p_0$. In stead of this two dimensional filtering, the local adaptive separable bilateral Gaussian filtering performs one dimensional bilateral filtering to the first dimension and performs the filtering again to the second dimension of the intermediate result [20]. SSPCSS executes blurring in the same way and removes light bleeding efficiently as shown in Fig. 4 (b). However, discontinuous penumbra occasionally appears near a complex occluder edge. This is an artifact caused by the approximation error of the dimensional decomposition because the filter is not essentially separable.

Since our extension replaces the two-dimensional Gaussian $g_f$ with an anisotropic Gaussian that contours an ellipse, which is inseparable, as explained in the last subsection, the approximation error increases.

The next subsection explains the filter decomposition and one more modification of Eq. (11), which makes the er-
The bilateral filters.

Based on the ratio of concerned areas gathered by the separation of these filtering results is expressed by Eq. (12), which is simply separate the filter as shown in Fig. 5(a). Therefore, SSABSS adopts a new filtering technique that mixes two filtering results by the two sequenced dimensionally decomposed filters. The first filtering performs one-dimensional filtering to a hard shadow image along with the minor axis of ellipses and then performs the same one-dimensional filtering along with the major axis of ellipses to the intermediate result by using the filter with minor axes. The second filtering performs the same process in reverse order. The mixing of these filtering results is expressed by Eq. (12), which is based on the ratio of concerned areas gathered by the separable bilateral filters.

\[
S_{dw}(p_0) = \frac{w_{sl}(p_0)f_{sl}(p_0) + w_{sh}(p_0)f_{sh}(p_0)}{w_{sl}(p_0) + w_{sh}(p_0)} = \frac{f_{sl}(p_0) + f_{sh}(p_0)}{w_{sl}(p_0) + w_{sh}(p_0)},
\]

where

\[
f_{sl}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p'')S_{dw}^{0}(p''),
\]

\[
f_{sh}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p'')S_{dw}^{0}(p''),
\]

\[
w_{sl}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p''),
\]

\[
w_{sh}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p''),
\]

\[
f_{sh}(p_1, p_2) = g_{sh}(p_2 - p_1)d_p(p_2 - p_1)
\]

The artifact in Fig. 5(a) is eliminated by the filtering as shown in Fig. 5(b).

\[
-f_{sh}(p_1, p_2) = g_{sh}(p_2 - p_1)d_p(p_2 - p_1),
\]

\[
f_{sh}(p_1, p_2) = g_{sh}(p_2 - p_1)d_p(p_2 - p_1) - \text{sgn}(A_{sh}(p_1) \cdot (p_2 - p_1))(p_2 - p_1)^{d_p(p_1)}
\]

\[
-f_{sh}(p_1, p_2) = g_{sh}(p_2 - p_1)d_p(p_2 - p_1) - \text{sgn}(A_{sh}(p_1) \cdot (p_2 - p_1))(p_2 - p_1)^{d_p(p_1)}
\]

\[
\text{sgn}(x) = \begin{cases} 
1, & x > 0 \\
0, & x = 0 \\
-1, & x < 0
\end{cases}
\]

3.3 Filter Decomposition

We decompose the bilateral filters introduced in the last subsection along with the minor and major axes of those ellipses. However, the filtering is an essentially inseparable process because weighting by similarity in a bilateral filter is inseparable, filers are not the same size and the axis directions are not constant. Clear artifacts may appear if we simply separate the filter as shown in Fig. 5(a). Therefore, SSABSS adopts a new filtering technique that mixes two filtering results by the two sequenced dimensionally decomposed filters. The first filtering performs one-dimensional filtering to a hard shadow image along with the minor axis of ellipses and then performs the same one-dimensional filtering along with the major axis of ellipses to the intermediate result by using the filter with minor axes. The second filtering performs the same process in reverse order. The mixing of these filtering results is expressed by Eq. (12), which is based on the ratio of concerned areas gathered by the separable bilateral filters.

\[
S_{dw}(p_0) = \frac{w_{sl}(p_0)f_{sl}(p_0) + w_{sh}(p_0)f_{sh}(p_0)}{w_{sl}(p_0) + w_{sh}(p_0)} = \frac{f_{sl}(p_0) + f_{sh}(p_0)}{w_{sl}(p_0) + w_{sh}(p_0)},
\]

where

\[
f_{sl}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p'')S_{dw}^{0}(p''),
\]

\[
f_{sh}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p'')S_{dw}^{0}(p''),
\]

\[
w_{sl}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p''),
\]

\[
w_{sh}(p) = \sum_{p' \in \ell_i(p)} f_{s}(p, p') \sum_{p'' \in \ell_j(p')} f_{s}(p', p''),
\]

\[
f_{sh}(p_1, p_2) = g_{sh}(p_2 - p_1)d_p(p_2 - p_1)
\]

4. Implementation

SSABSS is performed in four steps as shown in Fig. 1. In the figure, one area enclosed in an orange-lined box is executed by a shader process on GPU. The first step is off-screen rendering for a scene from the light position to obtain a shadow map that is a depth map from light position. The second step is rendering a scene from a camera position, which generates many buffers for hard shadows, penumbra size, depth, filter axes, material color, and shade on screen space. These two steps include 3D geometry calculations, whereas the other steps are just calculations of on screen space. The third step, which is the execution of Eq. (12), consists of three shaders, though four shaders are necessary to implement the equation in a straightforward way. This is because the second kernels of the filtering of the first sequence are the same as the first kernels of the filtering of the second sequence. This technique successfully reduces the increase in computational time. The final step, which combines material color, shade, and shadow by Eq. (1), executes pixels just one by one.

5. Results

Figure 6 compares SSABSS and SSPCSS. There is a bright area between two shadows when viewing the shadows from above (top row). When viewing from a more horizontal angle (bottom row), the bright area does not remain in the result of SSPCSS but does in the result of SSABSS. Since the objects and the dominant light in the scene are fixed, shadows should not change those shapes on the ground plane. Therefore, the result of SSABSS is more accurate than that
of SSPCSS. This is achieved by considering the relationship between the view direction and normals on shadowed surfaces.

Figure 7 compares an approximated ground truth, PCSS, SSPCSS, and SSABSS. As an ideal soft shadow, 400 point light sources that form a square area were prepared to emulate an area light source (a). In this area light and occluder layout conditions, the area for searching for occluders and blurring on the shadow map matches well and PCSS generates mostly correct shadow. However, the frame rate is about half that of SSABSS and SSPCSS. The shadow generated by SSPCSS is much too soft compared with the ideal shadow, because the filter kernels are always isotropic. Compared with SSPCSS, SSABSS generates a shadow more similar to that of PCSS, but it is also smoother. While the ideal penumbra’s intensity changes linearly and there is a clear edge between umbra and penumbra, penumbra’s intensity by our method does not change linearly and there is no clear edge. This is a limitation to generate a soft shadow lit by a non-rounded area light, so SSABSS uses Gaussian based blurring kernels. However, the error is difficult to notice if we do not know the shape of the area light, especially the occluders that have a more complex scene such as in Fig. 8. Therefore, we think the shadow of SSABSS has high enough visual quality.

Figure 9 shows the graph of the rendering time for the scene in Fig. 7. The horizontal axis corresponds to the size of the area light, $w_l$, used in Eq. (2). Since both SSABSS (yellow line) and SSPCSS (red line) use separable filtering, their graphs have similar tendencies.

Our method can be divided into four stages as shown...
in Fig. 1. We measured the execution time for every stage with the scene that 87 Stanford bunnies are placed, as shown in Fig. 10. Replacing different level of detailed bunnies, 4 conditions with different number of polygons are prepared. The measurement results are shown in Table 1 and Fig. 11.

Shadow map generation time depends on the total polygon numbers just like the other shadow map based methods. The step for shadow mapping, penumbra size estimation, and color and normal rendering also depends on the total polygon numbers. The number of samplings for blocker search is fixed in this measurement. However, since the program used for measurement is not optimized, the room of improvement in the speed remains. It is confirmed that the time of the blurring step does not depend on the polygon numbers. This is because it is processing on screen space. The last composition stage is performed in almost constant time.

These results show that the main blurring stages in SSABSS have relatively low cost calculation and do not depend on the scene complexity.

Figure 12 shows the soft shadows on the curved surfaces. The approximation of separable bilateral anisotropic Gaussian filters becomes less accurate as the axes change gradually. However, there is no fatal visual cue in the image.

6. Conclusion

In this paper, we introduced a new screen based soft shadow algorithm, SSABSS, which is based on blurring hard shadows with anisotropic essentially inseparable filters. However, adopting the new implementation of bilateral filtering, which mixes the results of two sequenced one-dimensional filterings, achieves a high efficiency method.

Though the implemented filters are essentially less accurate approximations, the fatal visual cue does not appear in the final image thanks to the new implementation of separable bilateral filtering.

The shadows rendered by SSABSS are more accurate than those of the previous method that works on screen space, SSPCSS, even though the computational cost is not much more. This efficiency is achieved by both the algorithm side and implementation side. The implementation is on GPU and works in a very high frame rate.
The shadow generated by SSABSS is softer than shadow of the ground truth with a square area light. This limitation of this method is due to it using Gaussian based kernels for blurring hard shadows. However, this difference is difficult to notice if there is no information about the shape of light. Thus, the visual quality of the results achieved by SSABSS is sufficient in various real time applications.

References

[1] M. MohammadBagher, J. Kautz, N. Holzschuch, and C. Soler, “Screen-space percentage-closer soft shadows,” ACM SIGGRAPH 2010 Posters, SIGGRAPH ’10, New York, NY, USA, p.133:1, 2010.

[2] Z. Zheng and S. Saito, “Screen space anisotropic blurred soft shadows,” ACM SIGGRAPH 2011 Posters, SIGGRAPH ’11, New York, NY, USA, p.75:1, 2011.

[3] F.C. Crow, “Shadow algorithms for computer graphics,” SIGGRAPH Comput. Graph., vol.11, no.2, pp.242–248, July 1977.

[4] L. Williams, “Casting curved shadows on curved surfaces,” SIGGRAPH Comput. Graph., vol.12, no.3, pp.270–274, Aug. 1978.

[5] T. Akenine-Möller and U. Assarsson, “Approximate soft shadows on arbitrary surfaces using penumbra wedges,” Proc. 13th Eurographics Workshop on Rendering, EGRW ’02, Aire-la-Ville, Switzerland, pp.297–306, 2002.

[6] G.S. Johnson, W.A. Hunt, A. Hux, W.R. Mark, C.A. Burns, and S. Junkins, “Soft irregular shadow mapping: Fast, high-quality, and robust soft shadows,” Proc. 2009 Symposium on Interactive 3D Graphics and Games, I3D ’09, New York, NY, USA, pp.57–66, 2009.

[7] E. Sintorn, O. Olsson, and U. Assarsson, “An efficient alias-free shadow algorithm for opaque and transparent objects using per-triangle shadow volumes,” ACM Trans. Graph., vol.30, no.6, pp.153:1–153:10, Dec. 2011.

[8] W.T. Reeves, D.H. Salesin, and R.L. Cook, “Rendering antialiased shadows with depth maps,” SIGGRAPH Comput. Graph., vol.21, no.4, pp.283–291, Aug. 1987.

[9] W. Donnelly and A. Lauritzen, “Variance shadow maps,” Proc. 2006 Symposium on Interactive 3D Graphics and Games, I3D ’06, New York, NY, USA, pp.161–165, 2006.

[10] R. Fernando, “Percentage-closer soft shadows,” ACM SIGGRAPH 2005 Sketches, SIGGRAPH ’05, New York, NY, USA, 2005.

[11] A. Lauritzen, “Summed-area variance shadow maps,” in GPU Gems 3, ed. H. Nguyen, ch. 8, Addison-Wesley Professional, 2007.

[12] B. Yang, Z. Dong, J. Feng, H.P. Seidel, and J. Kautz, “Variance soft shadow mapping,” Computer Graphics Forum, vol.29, no.7, pp.2127–2134, 2010.

[13] L. Shen, J. Feng, and B. Yang, “Exponential soft shadow mapping,” Computer Graphics Forum, vol.32, no.4, pp.107–116, 2013.

[14] T. Annen, Z. Dong, T. Mertens, P. Bekkaert, H.P. Seidel, and J. Kautz, “Real-time, all-frequency shadows in dynamic scenes,” ACM Trans. Graph., vol.27, no.3, pp.34:1–34:8, Aug. 2008.

[15] T. Annen, T. Mertens, P. Bekkaert, H.P. Seidel, and J. Kautz, “Convolution shadow maps,” Proc. 18th Eurographics Conference on Rendering Techniques, EGSR ’07, Aire-la-Ville, Switzerland, pp.51–60, 2007.

[16] G. Guennebaud, L. Barthe, and M. Paulin, “Real-time soft shadow mapping by backprojection,” Proc. 17th Eurographics Conference on Rendering Techniques, EGSR ’06, Aire-la-Ville, Switzerland, pp.227–234, 2006.

[17] B. Yang, J. Feng, G. Guennebaud, and X. Liu, “Packet-based hierarchical soft shadow mapping,” Computer Graphics Forum, vol.28, no.4, pp.1121–1130, 2009.

[18] L. Shen, G. Guennebaud, B. Yang, and J. Feng, “Predicted virtual soft shadow maps with high quality filtering,” Computer Graphics Forum, vol.30, no.2, pp.493–502, 2011.

[19] A. Robison and P. Shirley, “Image space gathering,” Proc. Conference