LETTER

An Efficient Soft Shadow Mapping for Area Lights in Various Shapes and Colors

Youngjae CHUN[†a)], *Nonmember and* **Kyoungsu OH**[†b)], *Member*

SUMMARY Shadow is an important effect that makes virtual 3D scenes more realistic. In this paper, we propose a fast and correct soft shadow generation method for area lights of various shapes and colors. To conduct efficient as well as accurate visibility tests, we exploit the complexity of shadow and area light color.

key words: *soft shadow, area light, global illumination, real-time rendering, GPU acceleration*

1. Introduction

In this paper, we introduce an efficient and accurate soft shadow generation method for area lights that have various shapes and colors. The representation of shadows in a virtual world is very important in 3D computer graphics, and many methods have been studied. In particular, soft shadows under area lights are a must for realistic 3D scene rendering, though they require much computation time. However, it is difficult to apply these methods to real-time applications because of the cost.

We therefore propose an efficient and realistic soft shadow rendering method. In order to accelerate the visibility test between the pixels and the area light, we adaptively filter shadow test results in a shadow map space. In this work, shadow test results located around a certain pixel are used to approximate visibility tests. Previous methods only calculated the average visibility by filtering for soft shadow rendering. However, they cannot handle area lights that have various shapes and colors. The proposed method performs, for each area light, visibility tests as detailed as possible by subdividing a filter area to refer to the shadow test results in the sub-regions. In particular, we subdivid a filter area in an adaptive manner based on the idea that the penumbra is visually more important than the umbra, or fully illuminated. As a result, our method shows realistic soft shadow from a light source in various shapes and colors in real-time.

2. Related Work

Soft shadow is one of the most impressive effects when we use an area light. However, illumination with an area light requires a number of visibility tests. It cannot be used for real-time 3D applications such as games because the cost is considerable. Therefore, various methods have been researched to render soft shadow in real-time. These methods can be classified into filtering-based methods and projection-based methods according to their approach.

Filtering-based methods filter shadow map results in a region of interest to generate soft shadow [1]–[3]. Fernando [1] used the positions of the light source, occluders, and a receiver to estimate a width of the penumbra for every pixel. An approximated visibility is computed by using a filter that is directly proportional to the estimated penumbra size. The visibilities in the filter are averaged for the final shadowing. However, this method suffers from the same problem as most screen-space methods. Yang et al. [2] introduced a soft shadow generation method based on [1] and [4]. They improved the light leak problem in the previous approach [4] while they were representing soft shadow. Since Donnelly and Lauritzen [4] uses a standard shadow map, if the shadow varies in a filter area, visual artifacts can occur. In order to solve this problem, a filter area is adaptively subdivided according to the variation of shadow in the filter area. Annen et al. [3] pre-filtered a shadow map by estimating the results of the shadow map test with Fourier series, as their previous work [5] did. [5] focused on anti-aliasing, but [3] handled all-frequency shadow effects in real-time. For this, they exploited an average depth value of occluders in a filter area to compute an optimal filter size. Their method can show hard and soft shadow simultaneously with different filter sizes for every pixel. Because of Fourier series approximation, however, sometimes it makes visual errors look like ringing artifacts.

Guennebaud et al. also used texels in a shadow map as potential occluders [6]. They assumed that the shadow map was parallel to an area light that looks like a white rectangle. Texels around the current pixel are projected onto a surface of the area light when the texel is closer to the light source. This means that the area of each texel projected is obscuring the area light for the current pixel. After projecting all occluder texels, they computed the ratio of the uncovered area to the whole area of the area light surface. The ratio was considered as a result of the visibility test. Soft shadow for an area light with arbitrary shape and color can be generated by this method. However, overlapping errors can occur when texels are projected onto the area light and it causes incorrect shadows. Atty et alused a special buffer called an occluder map to store the informa-

Manuscript received May 10, 2016.
Manuscript revised October 4, 2016.
Manuscript publicized November 11, 2016.

[†]The authors are with the Department of Media, Soongsil University, Korea.
a) E-mail: dkreformer@ssu.ac.kr
b) E-mail: oks@ssu.ac.kr

DOI: 10.1587/transinf.2016EDL8102

tion of occluders [7]. Every texel drawn in the occluder map was reconstructed as a micro-patch that is parallel to a light source. These patches are projected onto a surface of the light and accumulated to form a soft shadow map. Since the soft shadow map stores final shadow values, the researchers use just the map to present shadow. However, it is difficult to apply this method to more general cases because the occluders are separated from the entire scene. They play a role as only occluders, so there is no self-shadow on the occluders. In addition, their approach can suffer from perspective and projective artifacts.

3. Algorithms and Implementation

Our goal is to generate realistic soft shadow in real time for any area light that has arbitrary shape and color. We can present soft shadow in the left image of Fig. 1 by filtering shadow test results in a shadow map space. However, if we use only the average shadow test result then it might ingore various colors in the area light source. It cannot deal with more accurate illuminations shown in the right image of Fig. 1. The dotted lines indicate that some points are not seen by a part of the area light when physically correct visibilities are considered. In order to improve soft shadow quality, we adaptively subdivide the filter area and refer to more shadow test results.

The proposed method consists of filter size computationl, adaptive filter subdivision and final rendering. The first step is to compute an optimal size of a filter for every pixel. Similar to Convolution Shadow Maps [5], we define a cone as a target pixel and 4 corner vertices on the area light to find the intersection between the cone and a shadow map. The intersection on the shadow map is the initial filter area. The initial filter size can be optimized by using the average depth value of occluders in the filter area, as mentioned in Annen's previous work [3]. Each pixel has its own optimized filter, which allows us to generatet all-frequency shadow effects by averaging visibility values. The left image of Fig. 1 is the result of the above step. We can get more accurate soft shadow like as the right image in Fig. 1 if we use more specific visibilities in the filter area. Although we use optimally adjusted filters, it is still inefficient to refer to all locations in the filters.

For efficiency, we adaptively subdivide an optimal filter area according to the complexity of shadow in the filter area. If there is a significant change of shadow, we need to consider performing visibility tests for more positions in the filter region. We figured out that there are more complicated shadows in the penumbra rather than in the umbra or fully illuminated area. The shadow complexity can be determined by computing the variance of visibility test results. We commonly use the following equation to calculate variance:

$$V(x) = E(x^2) - E(x)^2 = E(x) - E(x)^2 \quad (1)$$

In the Eq. (1), $E(x)$ is the average visibility in the filter area and $V(x)$ is the variance of the visibility. Since visibility must be either 0 or 1, we don't have to store $E(x^2)$. Donnelly and Lauritzen [4] used an additional texture to store a square of depth value per each texel to compute the variance efficiently on GPU. In our case, we don't use the additional storage so that we can save memory and reduce the number of texture accesses.

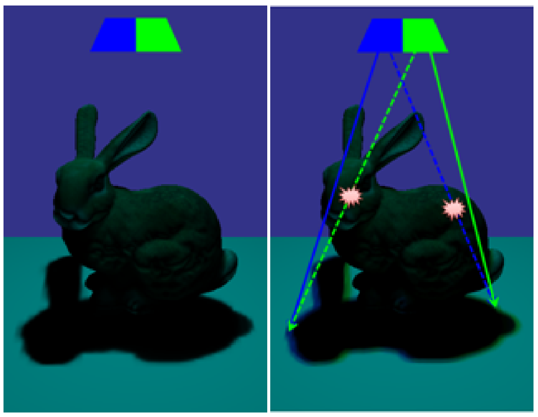

Fig. 1 Soft shadow generated by Convolution Soft Shadow Maps (left) [3] and by the proposed method (right)

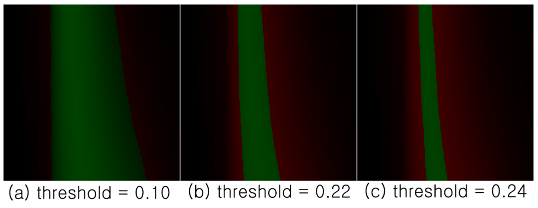

Fig. 2 Various threshold values for subdivision

In the case of umbra of fully illuminated region, the variance computed would be 0, so we do not subdivide the filter area in any events. On the other hand, if the variance is greater than the user-defined threshold, we separate the filter area into 4 equal subsections, like a quad tree. The variance of visibility in our method ranges from 0.0 to 0.25. We define the threshold used for subdivision as 0.2244. Pixels that have a variance greater than the threshold are more near the middle of penumbra area so we subdivide more. Fig. 2 shows how different subdivision areas are determined by various threshold values. For example, we subdivide twice for the green area but once for the red one. Even though every variance value doesn't fulfill a condition mentioned above, our algorithm runs subdivision until the number of subsections becomes 4×4 (16), with consideration for time cost and relative error between subdivision steps. For the subdivision, we also take account of the color complexity on the area light surface. When we use a colorful area light, we subdivide the filter area even if the shadow is not very changeful. However, if the area has constant color such as completely white then we don't need to subdivide the filter area. It is possible to compute the variance of area light image in real-time on GPU by using the approach introduced by Donnelly and Lauritzen [4]. Our algorithm works flexibly according to the characteristics of the area light and

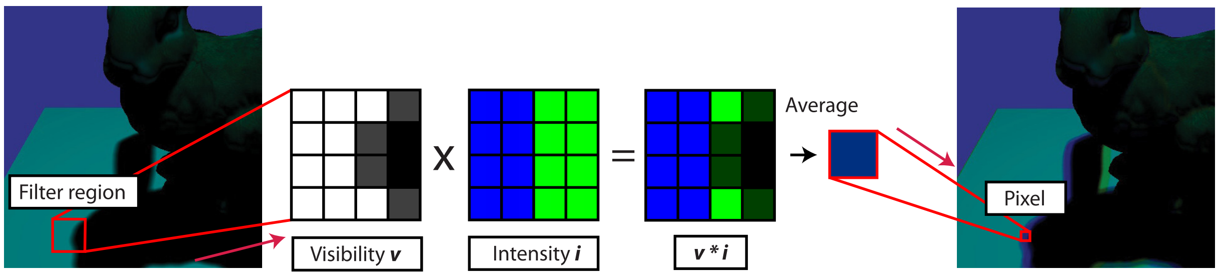

Fig. 3 An example of our soft shadow rendering for a pixel in the penumbra

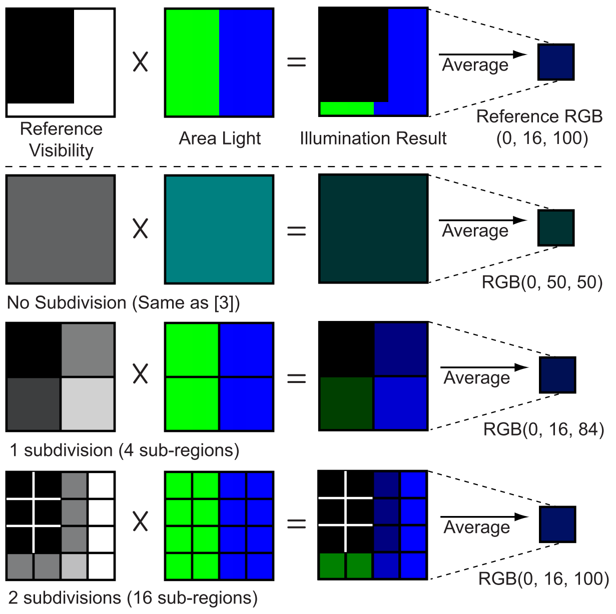
Reference Visibility
X
Area Light
=
Illumination Result
Average
Reference RGB (0, 16, 100)
No Subdivision (Same as [3])
RGB(0, 50, 50)
1 subdivision (4 sub-regions)
RGB(0, 16, 84)
2 subdivisions (16 sub-regions)
RGB(0, 16, 100)

Fig. 4 The reference illumination result and approximated results depending on the number of subdivisions

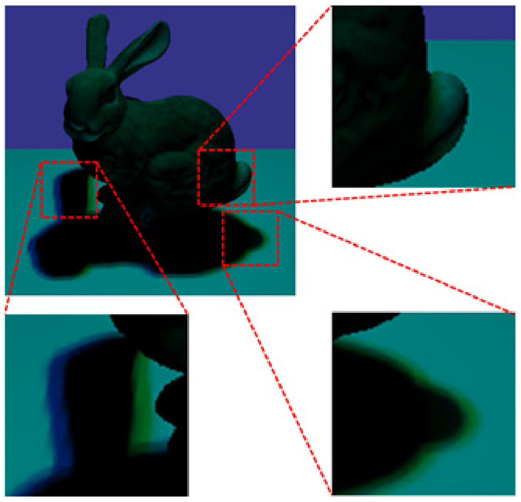

Fig. 5 A simple result and details

Table 1 Average frames per second (fps)

Subdivision Level	Convolution Soft Shadow Maps [3]	Our Method
Adaptive (0 ~ 2)	50fps	40fps
Fixed (2)		30fps

Table 2 The number of pixels in Fig. 5

Pixel type		The number of pixels (Ratio)
Not rendered (background)		243,260 (38%)
Rendered	No subdivision	367,460 (57.41%)
	1 subdivision	11,016 (1.73%)
	2 subdivisions	18,264 (2.86%)
Total (800*800)		640,000 (100%)

scene condition. It means that our method can use general and dynamic light sources, such as a candle or a torch, to generate realistic soft shadow.

After the subdivision, each visibility in the filter area is used for the visibility test of the corresponding partial surface of the area light. According to the visibility values each part of the area light is entirely or partly obscured. How to determine a pixel color in the penumbra through our method is shown in Fig. 3. First, we do texelwise multiplication of a visibility value in a filter and the corresponding intensity on the light source image. After then, we compute the average of results of multiplication for the final illumination. As a result, our algorithm draws realistic soft shadows that cannot be presented by previous methods that use an average visibility. Figure 4 shows various illumination results depending on the number of subdivisions. More subdivision gives us better result, but 4×4 (16) sub-filters are good enough for our purposes.

4. Experiments and Results

In this section, we talk about the quality and performance of our method. Figure 5 shows the primary visual effects and details generated by the proposed approach. Pixels in the detail view are in the shadow's penumbra, and our method accurately renders soft shadow for the pixels. More experiment results for the same scene with various area lights are shown in Fig. 6. A projection-based method [6] or a geometry-based method [8] presented realistic soft shadow for an area light, and our algorithm can show the same result in real-time but does not suffer from high computation cost or require additional geometry.

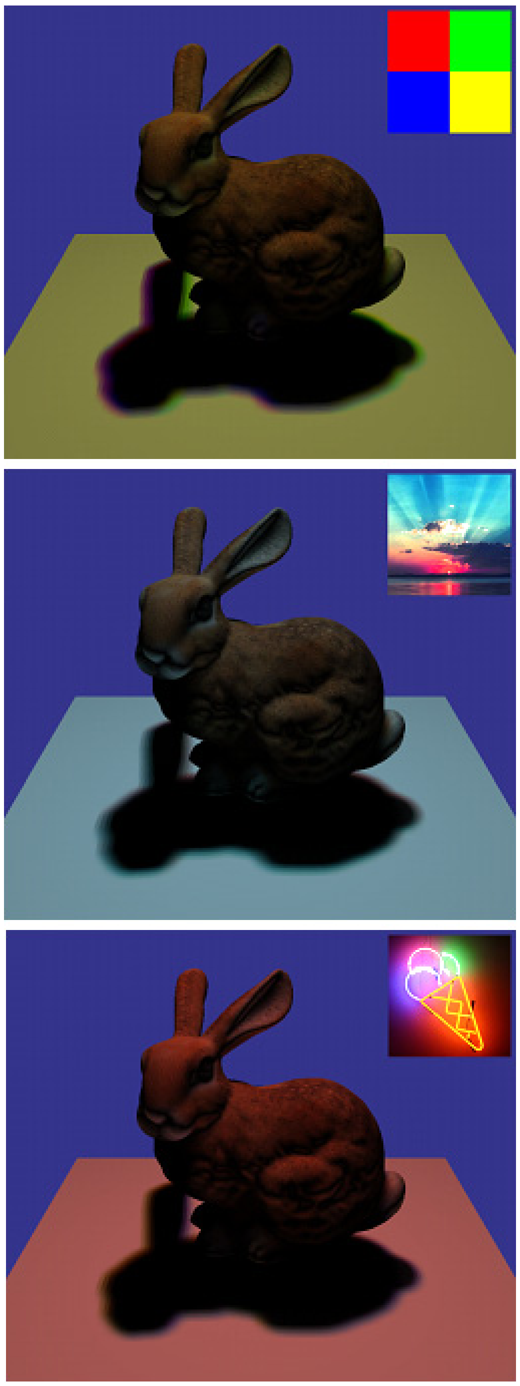

Fig. 6 Results with various area light images

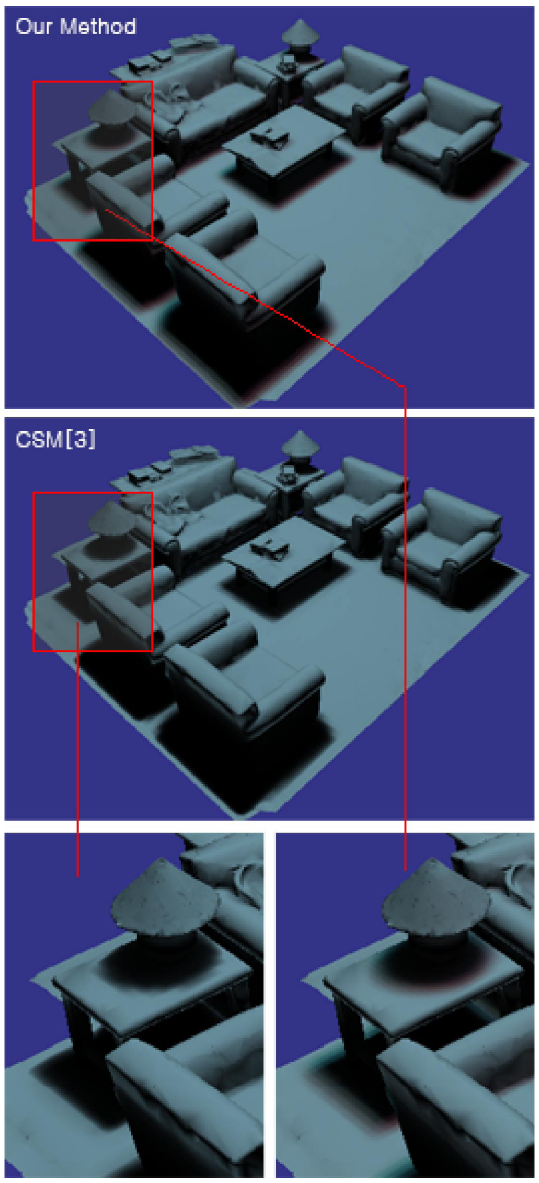

Fig. 7 A reading room scene

For the experiments, we used DirectX 3D 9.0 as a graphics library and NVIDIA GeForce GTX 460 as a graphics device. A 800×800 (px) resolution scene of about 40 frames per second was rendered using the proposed method (Table 1) where the number of triangles in the bunny model is 69,451. This method is little slower than the previous method that only uses an average visibility, but it can be applied to 3D applications. Though we set the number of subdivisions as 2 for every filter area, the algorithm still works in real-time.

In the main image in Fig. 5, the number of pixels that are treated for soft shadow in our algorithm is about 4.59% of the all pixels on the screen, as seen in Table 2. The number of pixels varies according to the position and the direction of the camera and light source but is still lower than the ratio of the number of pixels in the umbra or fully lit area generally. Taking care of the penumbra area is necessary to avoid significant visual artifacts, although the penumbra is relatively small.

Figure 7 and Fig. 8 show results for more general 3D scenes rendered by two methods. The reading room in the Fig. 7 is composed of 270,385 triangles and our method renders the scene at about 46 (adaptive)~36 (fixed) fps. One of the most significant visual differences between two methods are shown in each red rectangle. There are totally 211,609

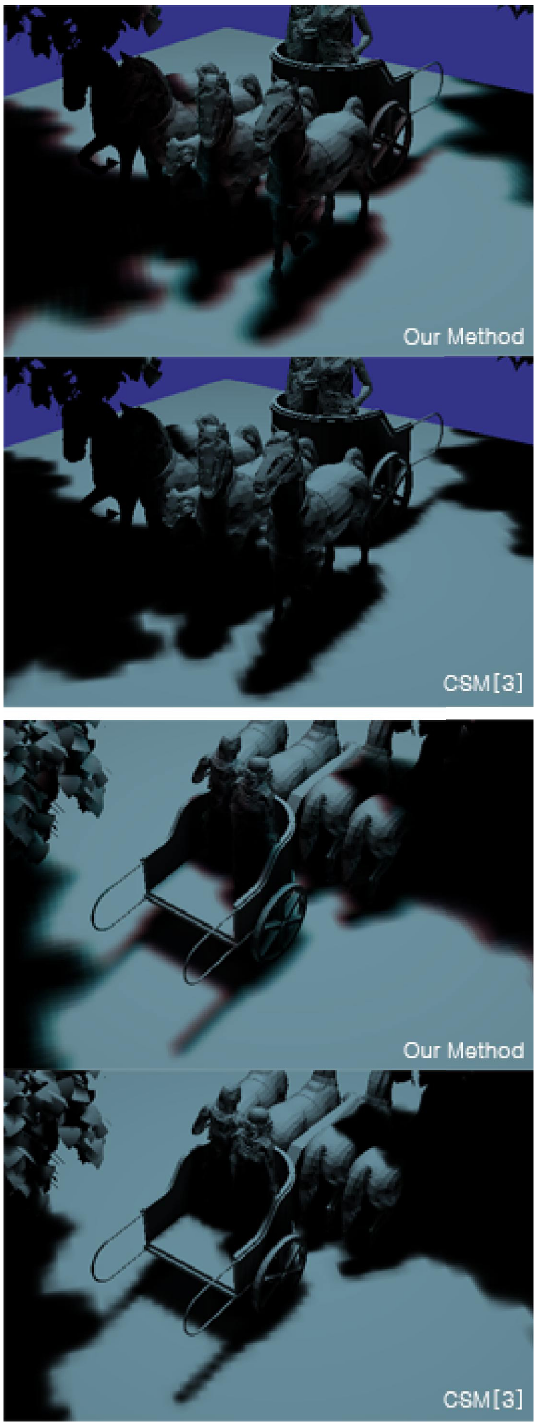

Fig. 8 A chariot and trees scene

triangles in a chariot and trees scene in the Fig. 8 and the average fps was about 35. It is easy to recognize clear improvement in soft shadow rendering in our method when more complicated 3D scenes are rendered.

5. Conclusion

In this paper, we introduced an efficient and accurate method for soft shadow rendering under area lights that have various shapes and colors. Soft shadow computation is known to be complicated and to take a considerable amount of time because of the number of visibility tests required. In order to improve performance, some approaches that use a shadow map to reduce the number of visibility tests have been proposed. Based on the previous methods, we can use an average visibility value in a certain region of interest per every pixel and represent soft shadow in real-time. However, these methods do not consider area lights in various shapes and colors, so they cannot be used for more general light sources.

We tried to improve visual quality by performing a visibility test as precisely as possible in the region of interest of a target pixel. For this, if there is a significant change of shadow around the target pixel, we separate the region of interest into four smaller, equal regions to refer to more specific visibility test results. For the subdivision, we also take account of the complexity of color variation in an image representing an area light. As a result, our method can generate realistic soft shadow correctly and fast. The proposed method shows more visually accurate soft shadow but provides almost the same performance as the previous soft shadow mapping methods. Furthermore, it also chooses rendering strategies that depend on how many times we subdivide the filter area according to the scene conditions or contents.

References

[1] R. Fernando, "Percentage-closer soft shadows," ACM SIGGRAPH 2005 Sketches, p.35, ACM, 2005.

[2] B. Yang, Z. Dong, J. Feng, H.-P. Seidel, and J. Kautz, "Variance soft shadow mapping," Comput. Graph. Forum, vol.29, no.7, pp.2127–2134, 2010.

[3] T. Annen, Z. Dong, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz, "Real-time, all-frequency shadows in dynamic scenes," ACM Trans. Graph., vol.27, no.3, p.34, 2008.

[4] W. Donnelly and A. Lauritzen, "Variance shadow maps," Proc. 2006 symposium on Interactive 3D graphics and games, pp.161–165, ACM, 2006.

[5] T. Annen, T. Mertens, P. Bekaert, H.P. Seidel, and J. Kautz, "Convolution shadow maps," Proc. 18th Eurographics Conference on Rendering Techniques, pp.51–60, 2007.

[6] G. Guennebaud, L. Barthe, and M. Paulin, "Real-time soft shadow mapping by backprojection.," Rendering Techniques, pp.227–234, 2006.

[7] L. Atty, N. Holzschuch, M. Lapierre, J.M. Hasenfratz, C. Hansen, and F.X. Sillion, "Soft shadow maps: Efficient sampling of light source visibility," omput. Graph. Forum, vol.25, no.4, pp.725–741, 2006.

[8] S. Laine, T. Aila, U. Assarsson, J. Lehtinen, and T. Akenine-Möller, "Soft shadow volumes for ray tracing," ACM SIGGRAPH 2005 Papers on - SIGGRAPH '05, vol.24, no.3, pp.1156–1165, 2005.