Revectorization-Based Shadow Mapping

Márcio C. F. Macedo[*] Antônio L. Apolinário Jr.[†]

Federal University of Bahia, Brazil

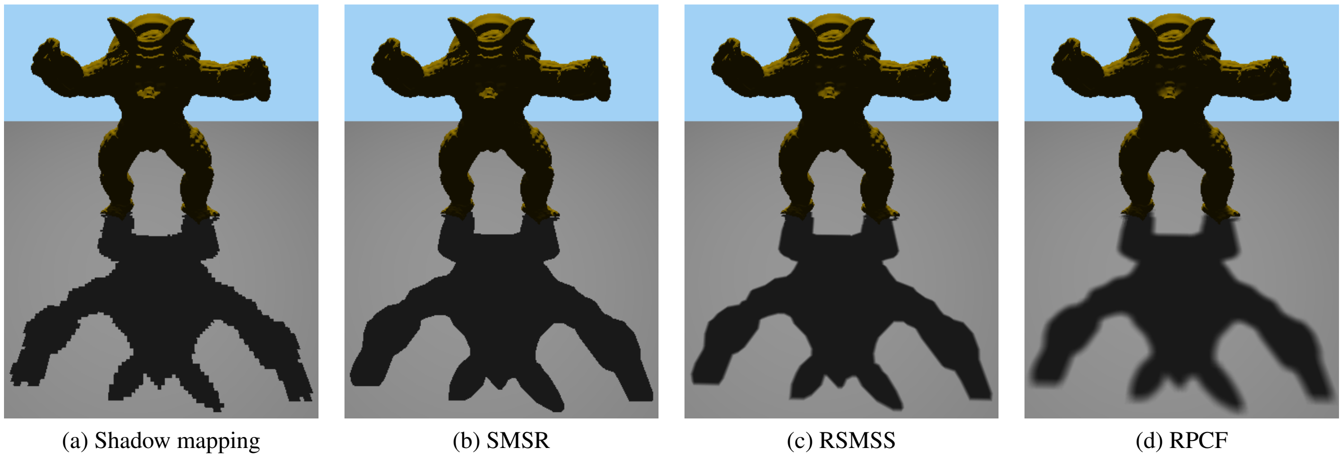

Figure 1: Comparison of shadow mapping (a) with the techniques proposed in this paper. Single-pass shadow map silhouette revectorization (SMSR) removes perspective aliasing by revectorizing the shadow boundaries (b). Revectorization-based shadow map silhouette smoothing (RSMSS) takes advantage of this revectorization to filter the anti-aliased hard shadows (c). Revectorization-based percentage-closer filtering (RPCF) is incorporated into the solution to control the filter kernel size (d). The image (d) was generated with the RPCF + RSMSS variant.

Abstract

Real-time rendering of high-quality, anti-aliased shadows is a challenging problem in shadow mapping. Filtering the shadow map reduces aliasing, but artifacts are still visible for low-resolution shadow maps or small kernel sizes. Moreover, the existing techniques suffer from light leaking artifacts. Shadow silhouette recovery reduces perspective aliasing at the cost of large memory footprint and high computational overhead for the shadow mapping. In this paper, we reduce aliasing with the revectorization-based shadow mapping. To effectively reduce the perspective aliasing, we revectorize shadow boundaries based on their discontinuity directions. Then, we take advantage of the discontinuity space to filter the shadow silhouettes, further suppressing the remaining artifacts. To control the filter kernel size, we incorporate percentage-closer filtering into the algorithm. This enables us to reduce jagged shadow boundaries, to simulate penumbra and to provide high-quality screen-space anti-aliasing. Compared to previous techniques, we show that shadow revectorization produces less artifacts, consumes less memory and offers real-time performance. The results show that our solution can be used in games and other applications in which real-time, high-quality shadows are desirable.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

[*]e-mail: marciocfmacedo@gmail.com
[†]e-mail: apolinario@dcc.ufba.br

1 Introduction

The faithful rendering of shadows is essential to improve the understanding and to provide realism for computer-generated scenes. However, the real-time rendering of high-quality shadows is still a challenging problem, mostly because computing photorealistic shadows still takes too much processing time to be done interactively for dynamic scenes.

Shadow mapping [21] is one of the most common algorithms used to compute hard shadows in real-time applications. It has several advantages, such as: simplicity, flexibility, scalability and hardware support. However, as an image-based approach, the finite resolution of the shadow map produces aliasing artifacts along shadow boundaries, false self-shadowing, and temporal incoherence [7].

Many approaches have been proposed to solve the problem of aliasing in shadow mapping. Filtering the shadow map is useful to reduce aliasing and to fake soft shadows, but the jagged shadow boundaries are still visible for shadow maps of low resolution or small kernel sizes. Also, most of the existing techniques suffer from light leaking, artifact in which a fully shadowed region is erroneously rendered as a lit region. As an alternative, shadow silhouette recovery is commonly used to compute accurate hard shadows. Unfortunately, this kind of techniques typically involves high memory and computational costs to reduce the perspective aliasing, which makes the methods unsuitable for real-time applications [19, 4, 9].

In this paper, we introduce the revectorization-based shadow mapping (RBSM). Shadow silhouettes are determined based on a discontinuity space. In this space, shadow edges are represented by their discontinuity directions. We take advantage of these discontinuity directions to revectorize and soften the shadow boundaries. For revectorization, we use a new variant of the shadow map silhouette revectorization (SMSR) algorithm [3]. To control the level of smoothness for the filtering, the percentage-closer filtering (PCF) algorithm [17] is incorporated into our solution.

In this context, our main **contributions** are: **1.** A shadow revectorization pipeline, which supports entering and exiting discontinuities. **2.** A variant of the SMSR algorithm which runs on the shader in a single pass, produces less artifacts and consumes less memory than the original approach. **3.** A filtering technique which takes advantage of a discontinuity space to effectively smooth shadow boundaries, producing less artifacts and consuming less memory than related work, while still providing real-time performance. **4.** A variant of the PCF algorithm which works on the filtered shadow silhouettes to reduce the perspective and banding artifacts of the original technique with smaller filter sizes and at higher frame rates.

2 Related Work

One of the first techniques proposed for real-time shadow rendering is the shadow mapping [21]. In this method, the scene is rendered from the light's point of view and the depth of the closest surface seen from the light is stored in a shadow map. Then, the scene is rendered from the camera viewpoint and the current depth values are compared to the depth values stored in the shadow map to determine whether a point in the scene is in shadow or not (i.e., shadow test).

Over the past decades, many approaches have been proposed to solve the main problem of shadow mapping: the aliasing. In this section, we review the most relevant methods related to our solution. A complete review of the existing shadow mapping algorithms is beyond the scope of this paper. The reader should refer to [7, 22]. Here, we classify the approaches based on their main strategy to alleviate aliasing in hard shadow mapping: filtering and silhouette recovery.

Filtering: Techniques of this category solve the problem of aliasing by smoothing the shadow silhouettes, resulting in the generation of soft, anti-aliased shadow edges. In shadow mapping, shadow maps cannot be filtered as conventional color maps because of the shadow test, which would be affected by the filtered depth values. Percentage-closer filtering [17] solves this problem by reversing the order of the filtering. First, the shadow test is performed, and then the filtering takes place, averaging the result of the depth comparisons over a filter region. PCF is easy to implement and provides good anti-aliasing, however it is prone to banding artifacts, does not support pre-filtering and is not scalable for large filter sizes.

To enable shadow map pre-filtering, the shadow test is commonly approximated by a filterable function. Variance shadow mapping (VSM) [6] and its variants [11, 12] use the Chebyshev's inequality approximation. Convolution shadow mapping (CSM) [1] uses the Fourier series to approximate the shadow test. In exponential shadow mapping (ESM) [2, 18], an exponential function is used. In exponential variance shadow mapping (EVSM) [12], an exponentially-warped VSM is used. Moment shadow mapping (MSM) [16] solves the shadow test according to the Hamburger moment problem. All of these methods are faster than PCF for large filter kernel sizes, produce pleasant visual results and are scalable. However, they usually suffer from light leaking artifacts in scenes with high depth complexity.

Filtering techniques are a good alternative to produce shadows which mimic the appearance of soft shadows. They avoid the complexity of soft shadowing and require low processing time. Furthermore, they contribute to the minimization of aliasing artifacts produced by shadow mapping. Unfortunately, these methods cannot remove the aliasing artifacts of low-resolution shadow maps for small filter sizes, as they work over jagged shadow boundaries. Increasing the resolution of the shadow maps may overcome this issue, at the cost of higher memory consumption and processing time. But, even in this case, any close-up on the shadow may reveal the aliasing artifacts. Larger kernel sizes remove the aliasing of the shadows, however, they severely blur out the shadows, losing too much detail of the shadow silhouette. Here, we propose a new technique which performs filtering over anti-aliased shadows. Our method does not generate perspective or banding artifacts even for small filter sizes, achieving high-quality anti-aliasing in real-time.

Silhouette Recovery: Rather than blurring the shadows, other techniques focus on shadow silhouette recovery to remove the jagged shadow edges. Instead of faking soft shadows, these techniques aim to compute hard shadows as accurate as the ones generated with shadow volumes [5] or ray tracing [20], however at high frame rates.

Hybrid approaches based on shadow mapping, shadow volumes [14, 4], and ray tracing [9] have been proposed in the literature. As a hybrid approach, these methods compute shadows faster than the reference solutions, but still much slower than shadow mapping.

To reconstruct accurate shadow silhouettes, some techniques rely on the storage of additional geometric information. In shadow silhouette mapping [19], the vertex which lies on the geometry silhouette is stored with the shadow depth map. Fast, sub-pixel antialiased shadow mapping [15] uses the pixel's position and associated face normal. Sub-pixel shadow mapping [13] stores triangle information (i.e., 3D vertex coordinates and depth derivatives) with the shadow map. Each one of these techniques has a different visibility function which uses this augmented information to reconstruct accurate shadow boundaries. However, they also increase memory consumption and processing time to achieve this goal.

Closest to our solution, shadow map silhouette revectorization [3] revectorizes shadow boundaries to reduce the perspective aliasing artifacts. Shadow edges are embedded into a discontinuity space and revectorized according to their discontinuity directions. The author suggests that the algorithm indeed alleviates aliasing, but the paper misses formalization and validation of the approach. Moreover, the method consists of two passes in the shader and does not work well for sloped surfaces, generating artifacts during revectorization.

Techniques for reconstructing shadow silhouettes are useful because they can solve the problem of aliasing in an accurate way. However, the existing techniques have large memory footprint or add high computational overhead to the shadow mapping. In this paper, we show that RBSM provides high-quality shadow silhouette recovery in real-time, while consuming as much memory as the traditional shadow mapping. Also, none of the existing shadow silhouette recovery techniques support filtering natively. In this paper, we propose a new solution to combine the strengths from both strategies (i.e., filtering and silhouette recovery) to effectively reduce perspective aliasing.

3 Revectorization-Based Shadow Mapping

In this section, we introduce the revectorization-based shadow mapping. Our work is mainly inspired by the two-pass shadow map silhouette revectorization algorithm described in [3]. In the first pass of this technique, jagged shadow edges are detected and stored in a discontinuity map. This texture stores color-coded values which indicate where the inner-side of the shadow edge is located. In the second pass of the method, similarly to morphological antialiasing [10], for each fragment belonging to a shadow edge, the distance between the fragment and both shadow edge ends is computed based on a traversal over the discontinuity map computed previously. This traversal results in the generation of the oriented normalized discontinuity space (ONDS). In this space, depending on the location of the fragment with respect to the shadow edge, it may or may not be revectorized by the algorithm.

In this paper, we propose several improvements over the approach presented in [3]: First, we propose a new revectorization algorithm which runs in a single pass on the shader (in addition to

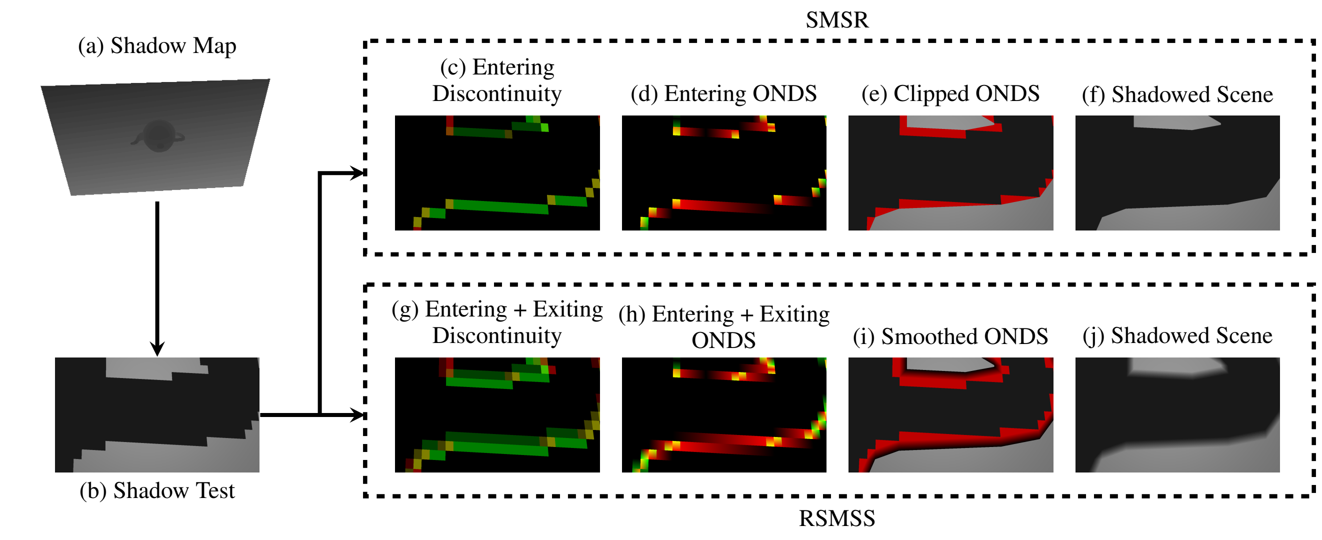

Figure 2: An overview of single-pass SMSR, and RSMSS techniques proposed in this paper. First, a shadow map is rendered from the light's point of view (a). Then, in the camera viewpoint, the shadow map is used to determine whether a fragment is in shadow or not (b). For the single-pass SMSR, entering discontinuities are computed in the exterior side of the shadow edges (c). Shadow boundaries are scanned according to the opposite axis of their discontinuity directions, resulting in the generation of an entering oriented normalized discontinuity space (ONDS) (d). By using an appropriate visibility function, ONDS is clipped (e), resulting in the generation of new shadows to be included into the final rendering (f). In RSMSS, exiting discontinuities are included into the pipeline (g-h). ONDS is smoothed, generating fake penumbras (i-j). The component $(d_c)_b$ was suppressed for visualization purposes.

the pass required for shadow map generation). In this case, we had to reformulate the algorithm to compute the ONDS because we do not generate a discontinuity map in a separate pass. We show that our algorithm is faster and consumes less memory than [3]. Second, we extend the algorithm to support entering and exiting discontinuities, located at the exterior and interior sides of the shadow silhouette edge, respectively. The approach proposed in [3] was designed to support only entering discontinuities during the ONDS computation. Third, a new approach was developed to reduce artifacts produced by the revectorization on sloped surfaces. Fourth, we integrate the native support for filtering into the revectorization pipeline through the technique named revectorization-based shadow map silhouette smoothing (RSMSS). In this method, we show that not only entering discontinuities, but also exiting discontinuities are useful to produce high-quality anti-aliasing. Finally, we propose a new variant of the PCF algorithm to control the level of smoothness in the final rendering. To achieve better performance and higher quality than the PCF algorithm, we incorporate the RSMSS into the PCF, reducing both perspective and banding artifacts of the technique using just a small kernel size.

RBSM can be used for both silhouette recovery and filtering. An overview of RBSM can be seen in Figure 2. Details can be found in the following subsections and in the supplementary document.

3.1 Revectorization Pipeline

To revectorize shadow boundaries in real-time, we must: render the shadow map, compute and normalize the discontinuity space, revectorize the shadow silhouettes and include them in the final shadowed scene. Each one of these steps is described in more detail below:

Shadow Map Rendering: First, we render the scene from the light's viewpoint and store the depth buffer in a shadow map (Figure 2-(a)).

Discontinuity Computation: Next, we compute the discontinuity for every texel projected in the camera viewpoint (Figures 2-(c), (g)). Let us denote z_l the depth stored in the shadow map and z_c the depth of the scene rendered from the camera viewpoint. The shadow test $s(x,y)$ for a texel (x,y) can be defined as a binary function that returns: 0 if $z_c > z_l$ (i.e., fragment is in shadow) and 1 otherwise (Figure 2-(b)). Also, let us define $n(x,y)$ the shadow test evaluation for a 4-connected neighbourhood: $n(x,y) = (s(x-o,y), s(x+o,y), s(x,y+o), s(x,y-o))$, where o is an offset equivalent to one shadow map sample.

We define discontinuity $d_u \in \{0,1\}$ as the absolute difference in the shadow test results for a texel and its 4-connected neighbours. Formally, $d_u = ||n-s||$. Therefore, the discontinuities are positive only at the shadow edges (i.e., where the shadow tests disagree) and zero elsewhere. If one desires to build a discontinuity map in an additional pass [10, 3], this definition of discontinuity would consume four color channels for each texel. To reduce memory consumption, the four-channel vector d_u is compressed into a three-channel vector d_c (Figures 2-(c), (g)) as follows:

$$\begin{aligned} (d_c)_{rg} &= \frac{2(d_u)_{rb} + (d_u)_{ga}}{4} \\ (d_c)_b &= 1 - s(x,y) \end{aligned}$$

	Discontinuity Components		
Value	$(d_c)_r$	$(d_c)_g$	$(d_c)_b$
0	No discontinuity	No discontinuity	Entering
0.25	Right	Top	–
0.5	Left	Bottom	–
0.75	Left and right	Top and bottom	–
1	–	–	Exiting

Table 1: The possible values for d_c and their meanings. The first two components of d_c store the discontinuity direction along horizontal ($(d_c)_r$) and vertical ($(d_c)_g$) axes. The component $(d_c)_b$ stores the type of the discontinuity.

As can be seen in Table 1, the components $(d_c)_r$ and $(d_c)_g$ are used to store the discontinuity direction (i.e., the direction where the shadow edge is located) along horizontal and vertical axes, respectively. The last term $(d_c)_b$ stores the type of discontinuity, which can be classified in entering or exiting. An entering discontinuity denotes that the current fragment is lit and the adjacent neighbour is shadowed (i.e., the current fragment is in the exterior part of the shadow silhouette). Similarly, an exiting discontinuity denotes that the current fragment is shadowed and the adjacent neighbour is lit (i.e., the current fragment is in the interior part of the shadow silhouette). To keep consistency with previous definitions which do not compute exiting discontinuities [3], we have defined $(d_c)_b = 0$ for an entering discontinuity and $(d_c)_b = 1$ otherwise.

Oriented Normalized Discontinuity Space Computation: In the discontinuity space, a jagged shadow edge is defined as an edge discontinuity which has beginning, end and length (Figures 3-(a), (b)). Each edge discontinuity has a dominant discontinuity, which is shared by all the texels belonging to the edge. For instance, in Figure 3-(a), the dominant discontinuity is to the top and in Figure 3-(b), it is to the bottom.

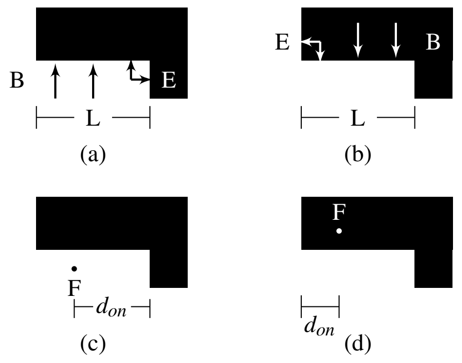

Figure 3: The properties of (a) entering and (b) exiting edge discontinuities: B - Discontinuity beginning. E - Discontinuity end. L - Discontinuity length. Arrow - Texel discontinuity (d_c). Black arrow - Entering discontinuity. White arrow - Exiting discontinuity. The relative position of the fragment F for entering (c) and exiting (d) edge discontinuities is defined by the parameter d_{on}, which is oriented towards the discontinuity end.

To compute the relative position $d_{on} \in [0,1]$ of the fragment inside the edge discontinuity (Figures 3-(c), (d)), we traverse the edge in the projected light space, since discontinuities are computed based on shadow map texels. Therefore, the step of the traversal is equivalent to one shadow map sample. The traversal is done for the two directions of the opposite axis of the dominant discontinuity (e.g., left and right directions for the cases shown in Figure 3) to find the edge discontinuity beginning and end. Details about the handling of special cases where the edge discontinuity does not have a beginning or end can be found in the supplementary document.

In our algorithm, the discontinuity beginning is found when the discontinuity directions of the current shadow map sample being traversed do not correspond to any discontinuity directions of the initial shadow map sample. The discontinuity end is found when the illumination condition goes from lit to unlit or vice-versa during the traversal (Figure 3-(a), (b)). We store the result of this traversal as a signed distance α between the texel and the end of the edge. In fact, α stores the number of steps taken during the traversal. We orient α such that it is positive if a discontinuity end has been found and negative otherwise.

Let us denote α_1 and α_2 the oriented distances computed for the two directions traversed and $L = |\alpha_1| + |\alpha_2| - 1$ the edge length. The oriented normalized discontinuity d_{on} is defined as:

$$d_{on} = (1 - \frac{\max(\alpha_1, \alpha_2)}{L}) + \frac{p_o}{L} \qquad (1)$$

where p_o is the relative position p of the fragment in relation to the projected shadow map sample, but oriented towards the discontinuity end. In practice, p is computed as the fractional part of the product between the 2-D coordinates of the pixel in the light space (normalized to $[0,1]$) and the shadow map resolution. Here, p_o is used to bring pixel-level accuracy for the ONDS computation. The subtraction in Equation (1) guarantees that the value of d_{on} grows towards the discontinuity end (i.e., d_{on} varies from 0 at the discontinuity beginning to 1 at the discontinuity end). Computing d_{on} for every fragment in the camera space results in the generation of the ONDS (Figure 2-(d), (h)).

To optimize performance and keep consistent frame rates, the user may fix the maximum number of steps given during traversal (i.e., the maximum edge length). In our tests, the use of a maximum of $L = 16$ was sufficient to keep consistent real-time performance and accurate ONDS computation.

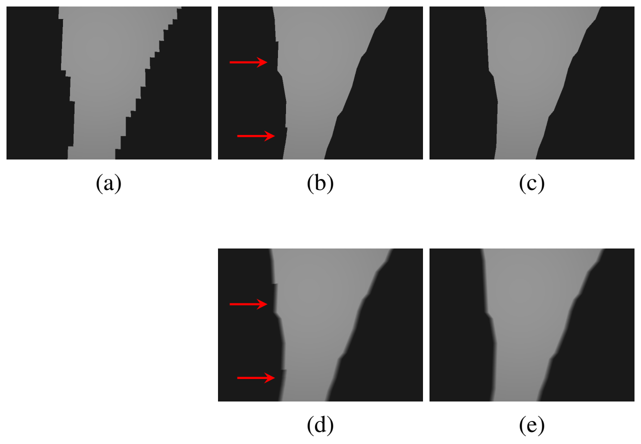

Figure 4: Jagged shadow edges (a) are revectorized with single-pass SMSR (b) and filtered with RSMSS (d). However, artifacts (pointed by red arrows) may arise for sloped surfaces due to the depth change, which affects the shadow test (b, d). By using our solution, we can solve this problem for both entering (c) and exiting (e) discontinuities.

For **sloped surfaces**, the depth of the shadow map sample being accessed may change in relation to the depth accessed in the initial shadow map texel. If we use the same z_c for shadow test during traversal, artifacts may arise in the cases where the depth change in z_l affects the shadow test result (Figures 4-(b), (d)). To solve this problem, we use the shadow mapping assumption that $z_c \geq z_l$ holds for every shadow map texel. Thus, to detect this depth change during traversal, we check whether $|z_c - z_l| < \varepsilon$ holds. If the condition is true, we update z_c before the shadow test: $z_c = z_c - \varepsilon$. In fact, we use a conservative approach which ensures that, for fragments in shadow, $z_c > z_l$. Conversely, for lit fragments, $z_c < z_l$. As shown in Figures 4-(c) and 4-(e), this solution provides good results for ONDS computation and alleviates the artifacts. To detect only the cases of depth change, the value of ε must be chosen carefully. In our setup, the difference between z_c and z_l is really small. Hence, we empirically have defined $\varepsilon = 2.5 \times 10^{-5}$, which has sufficed for all our test cases.

Oriented Normalized Discontinuity Space Clipping: In this step, we define a visibility function which works on the ONDS to compute the new shadow silhouettes (Figures 2-(e), (i)). The visibility function consists of a set of linear comparisons done over the oriented normalized discontinuity d_{on} and the sub-coordinates of the pixel in the light space p. As a general framework, the RBSM does not have a specific visibility function. This visibility function is specialized for each technique developed over RBSM, such as the single-pass SMSR and the RSMSS techniques. More details are described in the next subsections and in the supplementary document.

Final Rendering: Let us define $\phi(x,y)$ the shading function, $v(x,y)$ the revectorization-based visibility function, and $s(x,y)$ the hard shadow test. The final rendering function $f(x,y)$ can be expressed as $f(x,y) = \phi(x,y)s(x,y)v(x,y)$. By using this equation, we can render anti-aliased hard shadows, as depicted in Figures 2-(f), (j).

3.2 Single-Pass Shadow Map Silhouette Revectorization

The SMSR technique is a specialization of the RBSM which aims to revectorize the jagged shadow edges. In this method, we need to compute only entering discontinuities at the shadow silhouettes (Figure 2-(c)). Then, the discontinuity space is oriented and normalized only for entering discontinuities as well (Figure 2-(d)). We use the shadow mapping to classify fragments as lit or shadowed. Additionally, fragments that are back-facing the light source are classified as shadowed and do not participate in the discontinuity computation.

The SMSR visibility function $v_{\mathrm{SMSR}}(x,y)$ is a function which returns 0 if the fragment is a new shadow to be included after ONDS clipping and 1 otherwise (Figure 2-(e)). $v_{\mathrm{SMSR}}(x,y)$ consists of a set of 12 different shadowing configurations which group all the possible revectorization scenarios. A detailed description of this function is too long to be included here. We refer the reader to the supplementary material to find the full definition of $v_{\mathrm{SMSR}}(x,y)$.

By assuming $v(x,y) = v_{\mathrm{SMSR}}(x,y)$ in the final rendering function $f(x,y)$ (Section 3.1), we can generate anti-aliased hard shadows, as shown in Figures 1-(b) and 2-(f).

3.3 Revectorization-Based Shadow Map Silhouette Smoothing

The RSMSS technique uses the entering and exiting discontinuities of RBSM to add filtering for the anti-aliased hard shadows. RSMSS not only suppresses the aliasing artifacts to slightly lower frequencies, but also acts as a 1-D smoothing filter over the scene. This latter characteristic will be useful when extending this technique for 2-D filtering (Section 3.4).

In RSMSS, we compute entering and exiting discontinuities for all fragments in the camera viewpoint (Figure 2-(g)). ONDS is computed for entering and exiting discontinuities as well. Then, we define a visibility function $v_{\mathrm{RSMSS}}(x,y)$ which smoothes the ONDS to filter the anti-aliased hard shadows (Figure 2-(i)). Because we handle both entering and exiting discontinuities, we need a more complete visibility function to provide coherent filtering. Hence, $v_{\mathrm{RSMSS}}(x,y)$ consists of a set of 31 linear comparisons which return shadow intensities ranging in the interval $[0,1]$. Again, we refer the reader to the supplementary material to find a detailed description of $v_{\mathrm{RSMSS}}(x,y)$.

If we assume $v(x,y) = v_{\mathrm{RSMSS}}(x,y)$ in the final rendering function $f(x,y)$ (Section 3.1), we can render filtered hard shadows, as shown in Figures 1-(c) and 2-(j).

3.4 Revectorization-Based Percentage-Closer Filtering

As described in the previous subsection, the RSMSS technique produces filtered anti-aliased hard shadows. However, the filter size is fixed, proportional to the size of the shadow edge, which depends on the shadow map resolution. To enable control over the filter size, we incorporate PCF into the revectorization pipeline, creating the revectorization-based PCF (RPCF) technique.

The RPCF technique has two variants: The RPCF + SMSR variant uses the visibility function $v(x,y) = v_{\mathrm{SMSR}}(x,y)$ and provides filtering over the anti-aliased hard shadows. Unfortunately, similar to PCF, the RPCF + SMSR variant is still prone to banding artifacts and requires a high-order kernel to achieve good accuracy, which makes this technique unsuitable for real-time applications. To achieve high-quality anti-aliasing, reducing both banding and perspective artifacts for a small kernel size, we propose the RPCF + RSMSS variant, the incorporation of the 1-D RSMSS filter into the PCF, in which $v(x,y) = v_{\mathrm{RSMSS}}(x,y)$.

The evaluation of $v(x,y)$ for every texel inside the RPCF kernel is computationally expensive. Nevertheless, we can avoid the high cost of evaluating the visibility function for every sample. For the RPCF + SMSR variant, we only evaluate d_c for the samples in which $s(x,y) = 1$, because the SMSR technique handles only entering discontinuities. The visibility result for each neighbour sample $v_s(x,y)$ can be easily determined as follows:

$$v_s(x,y) = \begin{cases} 0 & \text{if } s(x,y) = 0, \\ 1 & \text{else if } d_c = 0, \\ v_{\mathrm{SMSR}}(x,y) & \text{otherwise.} \end{cases} \quad (2)$$

For the RPCF + RSMSS variant, we compute $v_s(x,y)$ as follows:

$$v_s(x,y) = \begin{cases} s(x,y) & \text{if } (d_c)_{rg} = 0, \\ v_{\mathrm{RSMSS}}(x,y) & \text{otherwise.} \end{cases} \quad (3)$$

In this case, for the texels where there is no discontinuity $(d_c)_{rg} = 0$, the visibility of the fragment is defined by the shadow test result. Otherwise, the RSMSS visibility function must be evaluated.

4 Results and Discussion

In this section, we evaluate the techniques in terms of visual quality, performance and memory consumption. We analyze the techniques for filtering and silhouette recovery separately. All images were generated using only one shadow map without any perspective optimization technique. In our experimental setup, memory and time usage was evaluated in an Intel Core$^{\mathrm{TM}}$ i7-3770K CPU (3.50 GHz), 8GB RAM, and an NVIDIA GeForce GTX 660 graphics card. All memory requirements were computed considering the mip-map overhead by a factor 1.3. ESM and EVSM are implemented without fallback to PCF and $c = 80$. For VSM, ESM, EVSM and MSM, a 3×3 two-pass separable Gaussian blur is used to filter the shadow maps. PCF and RPCF-based techniques were implemented with a simple 3×3 box filter (unless stated otherwise) to highlight the strengths and weaknesses for each technique. All the techniques are implemented with 16-bit quantization, with exception to ESM and EVSM, which require 32-bit [2, 12]. We use the stencil shadow volumes [8] for generating ground-truth hard shadows. To see the temporal consistency of RBSM and additional results of this work, we suggest the reader to see the accompanying video.

4.1 Rendering Quality

We compared the hard shadows computed from the single-pass shadow map silhouette revectorization technique with the ones produced by standard shadow mapping and shadow volumes, the latter being the ground-truth technique (Figure 5). We did not include the two-pass SMSR technique in this comparison because the visibility function of the original technique was not properly defined in the original paper [3]. In both scenarios, the closeups show that the SMSR technique recovers shadow boundaries at pixel-level (Figure 5-(b)), producing hard shadows which resemble the rendering quality of shadow volumes (Figure 5-(c)).

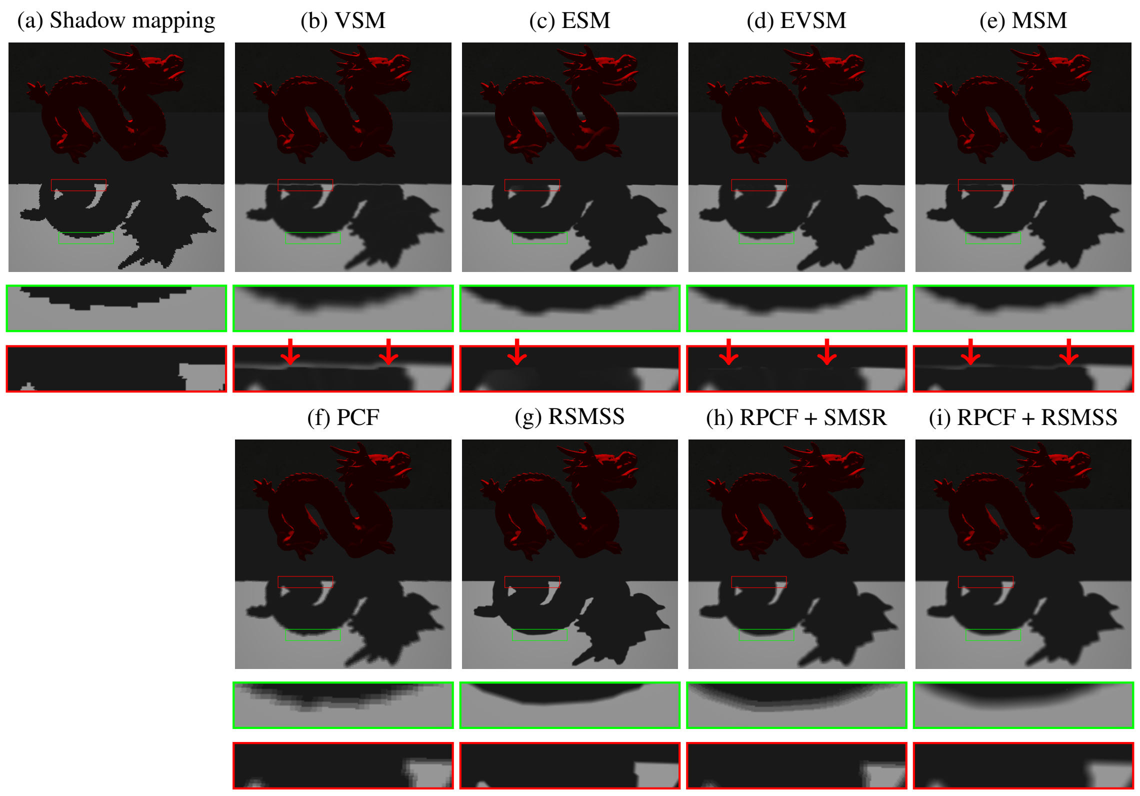

Figure 6: Filtered hard shadows produced by various techniques. Even in a simple scenario where the shadow of a wall is cast onto a dragon and a floor, several techniques suffer from light leaking artifacts (red arrows). Moreover, by blurring the aliased hard shadows, the filtered hard shadows still suffer from aliasing (green closeups). By using the techniques proposed in this paper, we can solve both problems. Images were generated for the Dragon model using a 1024^2 shadow map.

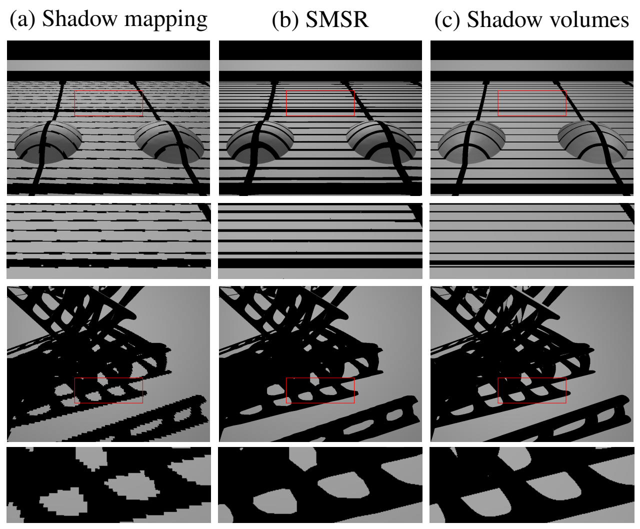

Figure 5: Hard shadows produced by shadow mapping, the SMSR technique and shadow volumes. Images were generated for Spheres (top) and YeahRight (bottom) models using a 2048^2 shadow map.

We visually compared the revectorization-based shadow map silhouette smoothing and the revectorization-based percentage-closer filtering techniques with related work, as shown in Figure 6. Shadow mapping suffers from aliasing (Figure 6-(a)). VSM, ESM and MSM techniques suffer from light leaking for the shadow cast by the wall onto the floor (Figures 6-(b, c, e)). Light leaking is greatly reduced in EVSM (Figure 6-(d)). However, for small kernel sizes, all these four filtering techniques do not remove the shadow mapping aliasing. PCF does not suffer from light leaking, but it does not remove the jagged appearance of the shadow edges for small kernel sizes (Figure 6-(f)). RSMSS is not prone to light leaking artifacts and effectively reduces the perspective aliasing. However, the filtering is performed in a limited extension (Figure 6-(g)). The RPCF + SMSR technique incorporates blurring into the revectorized shadows, but the method is still prone to banding artifacts (Figure 6-(h)). High accuracy is obtained for the RPCF + RSMSS technique, which takes advantage from the 1-D smoothing filter of the RSMSS to reduce both banding and perspective artifacts of the shadow filtering even for a low-order box filter kernel (Figure 6-(i)).

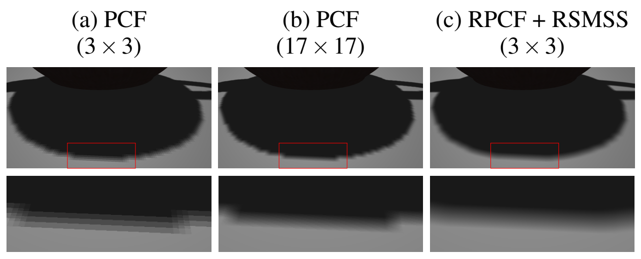

Figure 7: The influence of the kernel sizes over the filtered hard shadows produced by PCF and RPCF + RSMSS techniques. Images were generated for the Teapot model using a 1024^2 shadow map.

In Figure 7, we show a comparison between PCF and RPCF + RSMSS techniques in terms of filter size and rendering quality. As shown in Figure 7-(a), PCF suffers from banding artifacts for a low-order kernel size. Meanwhile, the RPCF + RSMSS algorithm requires a small filter size to provide high-quality anti-aliasing, reducing at the same time both banding and perspective artifacts (Figure 7-(c)). For the same filter size, RPCF + RSMSS is slower than PCF because filtering with revectorization demands increased processing time. To compare the visual quality achieved by both techniques for a similar performance, we must increase the filter size of the PCF technique to 17×17 (Table 5). In this case, the PCF effectively suppresses banding artifacts, but is still not able to reduce the perspective aliasing (Figure 7-(b)). This makes the RPCF + RSMSS a better choice for filtering, even for a 3×3 filter size.

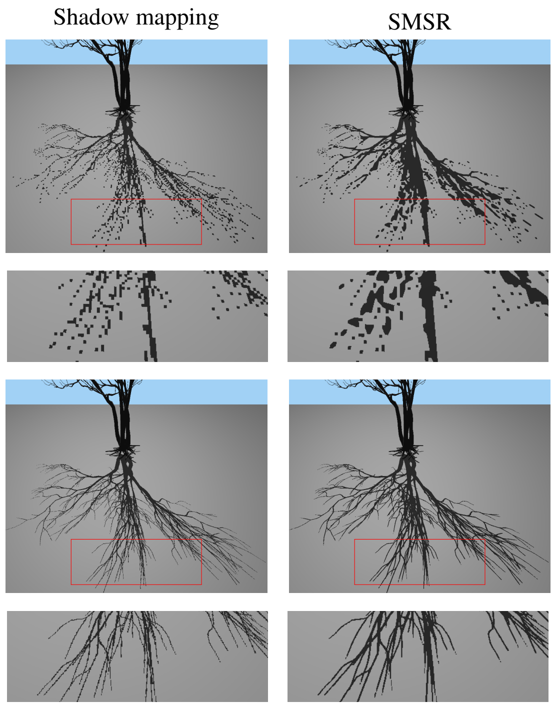

Figure 8: Standard shadow mapping and shadow map silhouette revectorization techniques applied for the Tree model for 1024^2 (top) and 4096^2 (bottom) shadow map resolutions.

Similar to related work (e.g., [13]), the quality of the revectorization-based shadow mapping is highly dependent on the shadow map resolution used. Because we do not rely on additional geometric details to perform the revectorization, we do not handle holes caused by the use of insufficient shadow map resolution (e.g., Figure 8, top). For models with fine details (e.g., the tree shown in Figure 8), revectorization does not perform well for low-resolution shadow maps. Increasing the resolution of the shadow map helps reducing the artifacts caused by the revectorization.

4.2 Performance

Tables 2 and 3 show timing results from the standard shadow mapping, our single-pass SMSR technique, the two-pass SMSR technique proposed in [3] and the shadow volumes. The two-pass SMSR is two times slower than the standard shadow mapping. Meanwhile, our single-pass SMSR adds an averaged computational overhead of 40% for a model with 100 000 polygons. Clearly, the single-pass SMSR technique outperforms the performance of the two-pass SMSR technique. Furthermore, the single-pass SMSR provides real-time performance, much faster than shadow volumes.

	Shadow Map Resolution			
Technique	512^2	1024^2	2048^2	4096^2
Shadow mapping	1.9	2.0	2.4	3.7
Single pass SMSR	2.4	2.6	3.3	5.9
Two pass SMSR	3.1	3.3	4.0	6.6
Shadow volumes	28.0	28.0	28.0	28.0

Table 2: Rendering times (in ms) for standard shadow mapping and the silhouette recovery techniques measured for the Dragon model ($\approx$ 100 000 polygons). Measurements include varying shadow map resolution.

	Output Resolution		
Technique	SD	HD	Full HD
Shadow mapping	1.9	2.0	2.3
Single pass SMSR	2.2	2.6	3.3
Two pass SMSR	2.9	3.3	4.0
Shadow volumes	25.0	28.0	40.0

Table 3: Rendering times (in ms) for standard shadow mapping and the silhouette recovery techniques measured for the Dragon model ($\approx$ 100 000 polygons). Measurements include varying output resolution. SD - Standard Definition ($480p$). HD - High Definition ($720p$). Full HD - Full High Definition ($1080p$).

The main advantage of our technique over the two-pass SMSR technique can be seen in Table 4. By running in a single-pass, our technique needs to render the scene from the camera viewpoint only once, which reduces processing time when rendering models with a moderate amount of polygons.

	Number of Polygons		
Technique	15 000	100 000	750 000
Shadow mapping	0.6	2.0	10.0
Single pass SMSR	0.9	2.6	10.1
Two pass SMSR	1.2	3.3	15.1
Shadow volumes	5.5	28.0	200.0

Table 4: Rendering times (in ms) for standard shadow mapping and the silhouette recovery techniques measured for the Teapot (15 000 polygons), Dragon (100 000 polygons) and YeahRight (750 000 polygons) models. Times were measured using a 1024^2 resolution shadow map.

Table 5 shows rendering performance for the shadow mapping and several filtering techniques. Shadow mapping and RSMSS were measured only for varying shadow map resolution. RSMSS is about two times slower than shadow mapping and it is slightly slower than related filtering techniques. The performance of RPCF-based filtering techniques is severely decreased for large filter sizes. However, as shown in Figure 7-(b), our solution is designed to provide high-quality anti-aliasing for small kernel sizes (e.g., 3×3 filter). In this sense, the cost of the use of the RSMSS technique as basis for the RPCF is compensated by the fact that the algorithm provides good results for small kernel sizes. PCF achieves near the same performance of the RPCF + RSMSS for a high-order kernel size of 17×17. In this case, the PCF technique is able to reduce banding artifacts, but it is still prone to perspective aliasing, both alleviated by the RPCF + RSMSS algorithm (Figure 7-(b, c)).

To achieve such performance results, we have tuned the revectorization-based shadow mapping to compute $f(x,y)$ efficiently. This optimization reduced $20-30\%$ of the computational cost of the RBSM-based techniques. More details about this tuning can be found in the supplementary document. Also, RBSM provides consistent real-time frame rates independent of lighting or viewing conditions, since the maximum number of steps for ONDS traversal is fixed. This makes our approach suitable for games and other applications in which consistent performance is essential.

		Shadow Map Resolution			
Filter Size	**Technique**	512^2	1024^2	2048^2	4096^2
	Shadow mapping	520	490	410	270
	RSMSS	260	240	210	145
3×3	PCF	470	445	410	270
	Pre-filtering	335	295	230	135
	RPCF + SMSR	180	165	145	110
	RPCF + RSMSS	90	85	80	70
10×10	PCF	200	180	150	90
	Pre-filtering	300	240	190	115
	RPCF + SMSR	40	36	32	30
	RPCF + RSMSS	13	13	13	13
17×17	PCF	85	80	70	40
	Pre-filtering	250	190	160	100
	RPCF + SMSR	20	16	15	13.5
	RPCF + RSMSS	5.5	5.5	5.5	5.5

Table 5: Performance table (measured in frames per second) for standard shadow mapping and the filtering techniques for the Dragon model. Pre-filtering represents the averaged performance of VSM, ESM, EVSM and MSM techniques. Measurements include varying shadow map resolutions and varying kernel sizes.

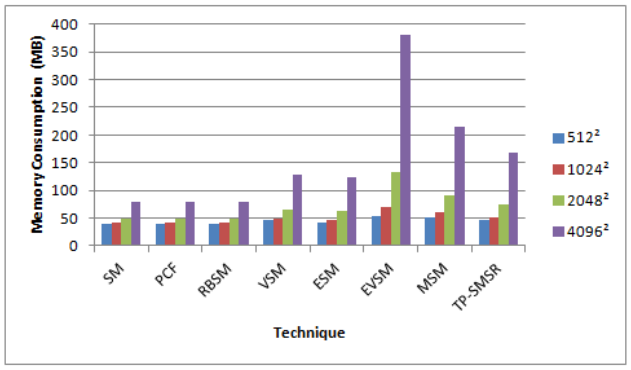

Figure 9: Memory consumption (measured in MB) for several shadowing techniques. Memory usage was measured for the YeahRight model. Measurements include varying shadow map resolutions. TP-SMSR: Two pass SMSR.

4.3 Memory Consumption

Figure 9 shows memory usage results for several shadowing techniques and shadow map resolutions.

Our revectorization-based techniques do not rely on any additional texture. In this sense, they use less memory than the two-pass shadow map silhouette revectorization, which stores d_c in a discontinuity map. Furthermore, we can see in Figure 9 that the proposed techniques provide the best memory results, comparable to shadow mapping. It is mainly because RBSM works with 16-bit quantization and needs only the shadow map texture (a single depth channel) to generate anti-aliased hard shadows.

5 Conclusion and Future Work

In this paper, we have proposed three revectorization techniques to reduce the aliasing artifacts in shadow mapping. The single-pass SMSR technique takes advantage of a discontinuity space to solve the problem of perspective aliasing in real-time. The RSMSS technique enhances anti-aliasing by filtering the anti-aliased hard shadows. The RPCF technique brings higher accuracy to PCF by incorporating it into RBSM. We have shown that our techniques are accurate, real-time, and do not suffer from light leaking artifacts, producing hard shadows that outperform state-of-the-art methods in terms of visual quality and memory consumption. By providing consistent, real-time frame rates, we believe that our approach is useful for games or other applications where perspective aliasing is still visible due to the use of low-resolution shadow maps or small kernel sizes for filtering. In this sense, the use of a small filter size to provide high-quality anti-aliasing makes the RPCF faster than the PCF to achieve an improved visual quality. We believe that the RPCF is ready to replace the PCF technique in applications which still use it for real-time hard shadow filtering.

Future work will extend the use of revectorization for other fields, such as soft shadows. In this case, strategies to reduce the computational cost of the soft shadow filtering, such as the restriction of the soft shadow calculation for fragments located in penumbra, will be useful to make the algorithm faster. Also, we believe that the RPCF will still require a low-order kernel size to provide high-quality soft shadow filtering. In terms of silhouette recovery, hybrid approaches which use revectorization with additional geometric details may be useful to improve the robustness of both techniques.

Acknowledgements

We are grateful to Vladimir Bondarev for discussing the ideas behind his original implementation of the two-pass shadow map silhouette revectorization method. Also, we would like to thank the anonymous reviewers for their insightful feedback. Armadillo and Dragon models are courtesy of Stanford Computer Graphics Laboratory. YeahRight model is courtesy of Keenan Crane. This research is financially supported by Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB) and Coordenação de Aperfeiçoamento de Pessoal do Nível Superior (CAPES).

References

[1] T. Annen, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz. Convolution shadow maps. In J. Kautz and S. Pattanaik, editors, *Rendering Techniques*, pages 51–60. The Eurographics Association, 2007.

[2] T. Annen, T. Mertens, H.-P. Seidel, E. Flerackers, and J. Kautz. Exponential shadow maps. GI '08, pages 155–161, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.

[3] V. Bondarev. Shadow map silhouette revectorization. I3D '14, pages 162–162, New York, NY, USA, 2014. ACM.

[4] E. Chan and F. Durand. An efficient hybrid shadow rendering algorithm. EGSR'04, pages 185–195, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[5] F. C. Crow. Shadow algorithms for computer graphics. SIGGRAPH '77, pages 242–248, New York, NY, USA, 1977. ACM.

[6] W. Donnelly and A. Lauritzen. Variance shadow maps. I3D '06, pages 161–165, New York, NY, USA, 2006. ACM.

[7] E. Eisemann, M. Schwarz, U. Assarsson, and M. Wimmer. *Real-Time Shadows*. A.K. Peters, 2011.

[8] T. Heidmann. Real shadows, real time. *Iris Universe*, 18:28–31, 1991.

[9] S. Hertel, K. Hormann, and R. Westermann. A hybrid GPU rendering pipeline for alias-free hard shadows. In *Proceedings of Eurographics 2009 Area*, pages 59–66, 2009.

[10] J. Jimenez, B. Masia, J. I. Echevarria, F. Navarro, and D. Gutierrez. Practical morphological anti-aliasing. In W. Engel, editor, *GPU Pro 2*, pages 95–113. AK Peters Ltd., 2011.

[11] A. Lauritzen. Summed-area variance shadow maps. In H. Nguyen, editor, *GPU Gems 3*, pages 157–182. Addison-Wesley, 2008.

[12] A. Lauritzen and M. McCool. Layered variance shadow maps. GI '08, pages 139–146, Toronto, Ont., Canada, 2008. Canadian Information Processing Society.

[13] P. Lecocq, J.-E. Marvie, G. Sourimant, and P. Gautron. Sub-pixel shadow mapping. I3D '14, pages 103–110, New York, NY, USA, 2014. ACM.

[14] M. D. McCool. Shadow volume reconstruction from depth maps. *ACM Trans. Graph.*, 19(1):1–26, Jan. 2000.

[15] M. Pan, R. Wang, W. Chen, K. Zhou, and H. Bao. Fast, sub-pixel antialiased shadow maps. *Computer Graphics Forum*, 28(7):1927–1934, 2009.

[16] C. Peters and R. Klein. Moment shadow mapping. I3D '15, pages 7–14, New York, NY, USA, 2015. ACM.

[17] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. SIGGRAPH '87, pages 283–291, New York, NY, USA, 1987. ACM.

[18] M. Salvi. Rendering filtered shadows with exponential shadow maps. In *ShaderX 6.0 Advanced Rendering Techniques*, pages 257–274. Charles River Media, 2008.

[19] P. Sen, M. Cammarano, and P. Hanrahan. Shadow silhouette maps. *ACM Trans. Graph.*, 22(3):521–526, July 2003.

[20] T. Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.

[21] L. Williams. Casting curved shadows on curved surfaces. SIGGRAPH '78, pages 270–274, New York, NY, USA, 1978. ACM.

[22] A. Woo and P. Poulin. *Shadow Algorithms Data Miner*. CRC Press, 2012.