Computers & Graphics (2017)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Improved anti-aliasing for Euclidean distance transform shadow mapping

Márcio Macedo[a,*], Antônio Apolinário[a]

[a] *Federal University of Bahia, Av. Ademar de Barros, 40.170-110, Bahia, Brazil*

ARTICLE INFO

Article history:
Received November 24, 2017

Keywords: Rendering, Real time, Shadows, Anti-Aliasing

ABSTRACT

High-quality, real-time penumbra rendering remains a challenging problem in computer graphics. Existing techniques for real-time fixed-size penumbra simulation generate aliasing, banding or leaking artifacts that diminish the realism of shadow rendering. Euclidean distance transform shadow mapping aims to solve that by using a normalized Euclidean distance transform to simulate penumbra on the basis of anti-aliased hard shadows generated by revectorization-based shadow mapping. Despite the high visual quality obtained with such a technique, the anti-aliasing provided by shadow revectorization comes at the cost of shadow overestimation artifacts that are introduced in the scene. In this paper, we propose an improved algorithm for Euclidean distance transform shadow mapping by reformulating the visibility function of revectorization-based shadow mapping. Through an additional detailed analysis of the results, we show that we are able to reduce shadow overestimation artifacts for penumbra simulation, generating shadows with higher visual quality than previous fixed-size penumbra shadowing methods, while keeping real-time performance for shadow rendering.

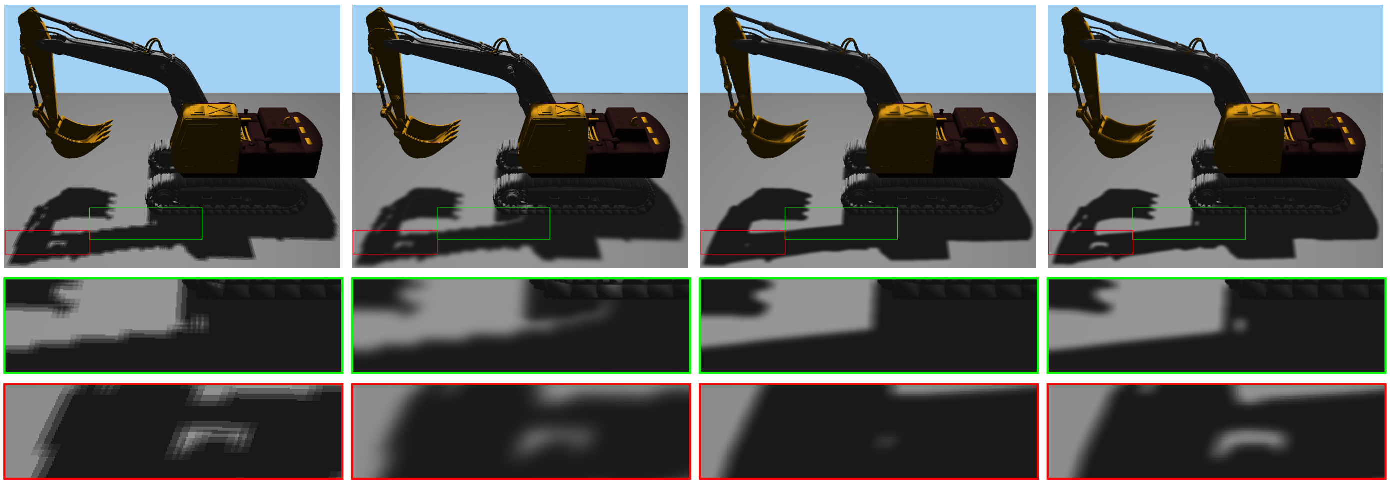
1. Introduction

Shadows are essential in several computer graphics applications, such as games and augmented reality, because they add a compelling effect, increasing the visual perception of the user with respect to the rendering of virtual scenes [1]. As pointed in [2], users usually prefer realistic shadows over fake ones when looking into virtual scenes. Unfortunately, accurate shadow rendering is still not feasible for real-time applications, mainly because a high number of samples must be taken from an area light source to approximate the direct illumination term of the rendering equation [3, 4], making the shadowing process costly.

One of the most traditional ways to compute shadows in real time is shadow mapping [5]. By approximating the area light source by a single point light source, this technique discretizes the 3D virtual scene, as seen from the point light source viewpoint, into a depth buffer named *shadow map* that is used to aid the real-time shadow computation. However, shadows generated on the basis of a shadow map are prone to aliasing artifacts and temporal incoherence due to the finite resolution of the shadow map. Moreover, differently from an area light source, a point light source is not able, in essence, to cast penumbra in the scene because this type of light source is infinitesimal, such that it cannot be partially occluded in the scene. Therefore, shadow mapping is only able to simulate hard shadows (i.e., shadows without the penumbra effect) in the scene. Unfortunately, such hard shadows are unrealistic, because they are not much present in the real world.

Aliasing artifacts are commonly suppressed by the use of texture linear filtering techniques, such as mip-mapping [6] and anisotropic filtering [7]. However, these strategies cannot be directly applied in the shadow map, because shadow mapping uses a non-linear shadow test to determine the visibility condition of a given fragment [8]. Then, several techniques have been proposed to allow shadow map filtering. Existing techniques

*Corresponding author
e-mail: marciocfmacedo@gmail.com (Márcio Macedo), antonio.apolinario@ufba.br (Antônio Apolinário)



(a) PCF (b) VSM (c) EDTSM (d) EDTSM*

Fig. 1. Fixed-size penumbra produced by different techniques. For a low-order filter size, shadow map filtering techniques, such as PCF, generate shadows with aliasing and banding artifacts (a). Shadow map pre-filtering techniques, such as VSM, are prone to light leaking artifacts (green closeup in (b)). EDTSM suffers from shadow overestimation artifacts (c). The proposed approach (here named EDTSM*) is able to minimize those artifacts efficiently (d). Images were generated for the Excavator model using a 512^2 shadow map resolution.

either realize shadow filtering after the shadow test [9, 10] or filter the shadow map (as done in [11, 12]), such that the shadows produced by a modified version of the shadow test are already filtered and anti-aliased. While these techniques minimize aliasing artifacts and simulate **fixed-size** penumbra, they introduce new artifacts in shadow rendering because of the filtering strategy used. Banding artifacts may appear in the final rendering if low-order filter sizes are used to keep real-time performance [9] (Figure 1-(a)). Techniques that filter the shadow map before the shadow test are prone to light leaking artifacts (in which a shadowed region is erroneously assumed as a lit region) because the filtering may incorrectly affect the shadow test result [13, 14] (green closeup in Figure 1-(b)). Techniques that filter the shadow map after the shadow test are prone to shadow overestimation artifacts because, during the shadow anti-aliasing, they can incorrectly merge parts of the shadow boundary that are originally disconnected [10] (Figure 1-(c)). Finally, filter size may directly affect the quality of the penumbra simulated. Small filter sizes may produce penumbra with blurred aliasing artifacts along the shadow boundary (Figure 1-(a)). On the other hand, large filter sizes may suppress fine details of shadows into penumbra.

Recently, Euclidean distance transform shadow mapping (EDTSM) was introduced to solve most of the problems mentioned before [15]. To do so, the technique first computes anti-aliased hard shadows using revectorization-based shadow mapping (RBSM) [10]. Then, an exact normalized EDT is computed from anti-aliased hard shadows using parallel banding algorithm (PBA) [16], which runs on the GPU. Finally, to reduce skeleton artifacts generated by the EDT, a simple mean filter is applied over the shadow boundary. Indeed, EDTSM is able to simulate fixed-size penumbra with less aliasing, banding and leaking artifacts than previous work, while keeping high frame rates. However, by the use of RBSM as hard shadow anti-aliasing technique, EDTSM suffers from shadow overestimation artifacts, which decrease the realism of shadow rendering (Figure 1-(c)).

In this work, which is an invited extension of our Graphics Interface 2017 paper [15], our main contribution is the enhancement of the RBSM visibility function to solve the problem of shadow overestimation, as shown in Figure 1-(d). Doing so, we can improve not only the quality of the hard shadow anti-aliasing provided by RBSM, but also the quality of the fixed-size penumbra simulation generated by EDTSM, keeping the processing time with a marginal overhead (about 1% of additional cost).

2. Related Work

In this section, we review relevant work related to the proposed solution. We mainly cover techniques which provide real-time fixed-size penumbra simulation. For a more complete review of existing shadow mapping techniques, we suggest the reader to see the following books [17, 18].

Several strategies have already been proposed to solve the aliasing problem of shadow mapping by warping [19, 20], partitioning [21, 22], traversing [23, 10] or incorporating additional geometric information into the shadow map [24, 25, 26]. Unfortunately, none of these strategies are able to simulate penumbra, focusing only on the anti-aliasing of hard shadows.

The most traditional algorithm for fixed-size penumbra simulation is the percentage-closer filtering (PCF) [9]. As an extension of shadow mapping, PCF takes the results of shadow tests performed over a filter region and averages them to determine the final shadow intensity. By filtering the shadow test results, rather than the shadow map itself, PCF is not prone to light leaking artifacts, but provides real-time performance, while keeping low memory consumption for penumbra simulation. However, PCF does not support texture pre-filtering, does not provide scalability in terms of filter size, and requires a high number of samples to solve banding artifacts.

To make the shadow filtering scalable, variance shadow mapping (VSM) [11] uses Chebyshev's inequality, depth and squared depth stored in the shadow map to determine the shadow intensity of a surface point by means of a probability of whether the point is in shadow. VSM supports shadow map pre-filtering and is scalable for the filter size, but generates light leaking artifacts in shadows.

To reduce the light leaking artifacts of VSM, convolution shadow mapping (CSM) [27] uses Fourier series to approximate and linearize the shadow test. In CSM, the shadow map is converted into filtered basis textures that are used to determine the final shadow intensity as a weighted sum of basis functions stored in basis textures. CSM supports pre-filtering and reduces light leaking artifacts as compared to VSM, at the cost of more memory consumption and processing time than VSM.

To minimize the processing time required by CSM, exponential shadow mapping (ESM) [12, 13] approximates the shadow test by an exponential function, rather than Fourier series. ESM stores exponent-transformed depth values into the shadow map, which are later used for penumbra simulation. ESM is faster and requires less memory footprint than CSM, while generating visual results similar to the ones obtained with VSM.

To improve the visual quality of both VSM and ESM, exponential variance shadow mapping (EVSM) [28] merges both ESM and VSM theories to produce high-quality fixed-size penumbra simulation. In EVSM, light leaking only occurs at places where both ESM and VSM techniques generate such an artifact.

As an alternative to both VSM and ESM techniques, Gaussian shadow mapping (GSM) [29, 30] replaces Chebyshev's inequality by a Gaussian cumulative distribution function to minimize light leaking. Also, inspired by EVSM, GSM warps its visibility function to take advantage of the exponential function proposed in ESM to further reduce light leaking.

Moment shadow mapping (MSM) [14, 31] improves VSM by storing four powers of depth in the shadow map and treating the penumbra simulation as a Hamburger or Hausdorff moment problem. MSM reduces the light leaking artifacts of VSM, generates results similar to EVSM, while keeping nearly the same rendering time of both techniques, but consuming more memory requirements than VSM.

The shadow filtering techniques presented in this section try to hide the aliasing artifacts generated by shadow mapping by the simulation of the penumbra effect. However, for small penumbra sizes, a high-order filter size must be used to suppress both aliasing and banding artifacts at the penumbra location. Taking advantage of both shadow anti-aliasing and fixed-size penumbra simulation strategies, the revectorization-based PCF (RPCF) [10] applies PCF over anti-aliased hard shadows generated with RBSM to simulate the penumbra effect. In fact, RPCF is able to generate high-quality fixed-size penumbra even for low-order filter sizes and low-resolution shadow maps. However, RPCF is slower than PCF and shadow map filtering techniques because of the additional cost of the shadow revectorization, which reduces its applicability in real-time applications.

To make RPCF more scalable and faster, EDTSM [15] estimates the penumbra intensity of a region by an exact normalized EDT that is computed over revectorized hard shadows. EDTSM is able to minimize aliasing, banding and light leaking artifacts, but suffers from the overestimation caused by the shadow revectorization.

To the best of our knowledge, the only existing solutions that make use of EDT to simulate penumbra are the stylized shadows [32] and our previous work, EDTSM [15]. Stylized shadows compute a signed distance function from an accurate hard shadow to generate artistic, non-real-time, non-photorealistic shadows. Distance transform is specially used to control shadow and variable-size penumbra sizes. EDTSM computes a normalized EDT from a revectorized hard shadow to simulate fixed-size penumbra in real time. In this technique, distance transform is used to compute the shadow intensities that make the normalized EDT to resemble a penumbra.

In general, penumbra simulation techniques are an efficient alternative to shadow mapping. Techniques that filter the shadow map typically warp the depth stored in the shadow map into another basis function to make the penumbra simulation scalable in terms of filter size. However, shadow map filtering introduces noticeable light leaking artifacts, which reduce the shadow visual quality. On the other hand, PCF, RPCF and EDTSM techniques filter the hard shadows produced with shadow mapping to avoid light leaking. However, PCF and RPCF are not scalable with respect to the filter size, while EDTSM suffers from overestimation artifacts. Since our goal is to simulate high-quality fixed-size penumbra in real time, we show how we can improve the visual quality of shadows generated with EDTSM by changing the RBSM visibility function, reducing shadow overestimation artifacts, while keeping real-time performance.

3. Euclidean Distance Transform Shadow Mapping

EDTSM is a technique that uses Euclidean distance transform to simulate the penumbra effect over anti-aliased hard shadows. The main assumption of EDTSM is that the penumbra intensity of a fragment can be approximated by the Euclidean distance of the fragment to the nearest fragment located in the hard shadow boundary. Then, this Euclidean distance is normalized inside a user-defined fixed-size penumbra region because the penumbra intensity of a fragment must lie in the interval of intensities between the umbra and lit regions.

As described above, EDTSM requires the use of a hard shadow anti-aliasing technique before the fixed-size penumbra simulation. In our previous attempt, we have used RBSM to perform the shadow anti-aliasing (Section 3.1). However, the original visibility function of RBSM suffers from shadow overestimation. In this work, we do propose an improved visibility function for RBSM to solve the shadow overestimation problem of RBSM and, consequently, EDTSM (Section 3.2).

3.1. Revectorization-Based Shadow Mapping

The first step of EDTSM is the generation of anti-aliased hard shadows. To do so, we make use of RBSM, an algorithm that

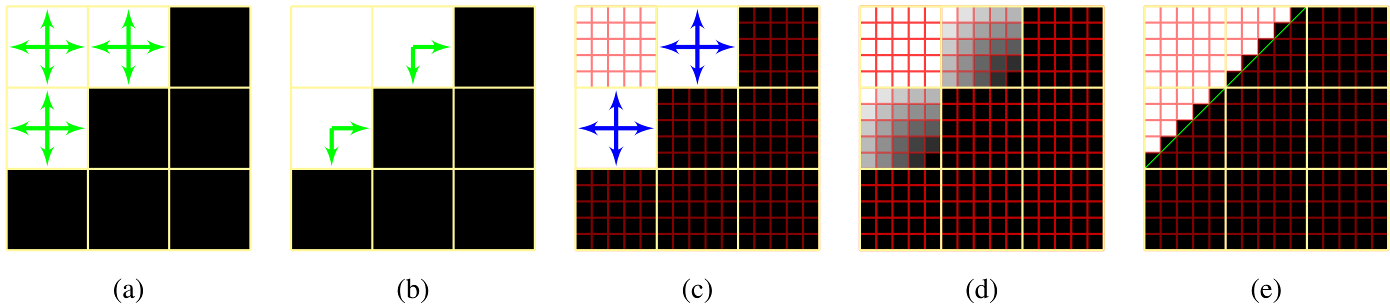


Fig. 2. An overview of the original RBSM. First, a shadow test is computed for every fragment projected in the light space (yellow grid). Then, for each lit fragment, a neighbourhood evaluation (green arrows in (a)) is conducted to detect the fragments located in the aliased shadow boundary (a). With the neighbourhood evaluation, we are able to not only detect aliased fragments, but also the directions of where the shadow boundary is located (b). Next, the algorithm traverses the light space to determine the size of the aliasing (blue arrows in (c)) and the normalized relative distance of each fragment located in the shadow boundary to the origin of the local aliasing (d). Finally, a visibility function is used to define the new anti-aliased shadow boundary in the camera space (red grid) (e).

aims to minimize the hard shadow aliasing problem by taking advantage of the increased screen-space resolution provided by the camera view to traverse the shadow boundary, recovering an approximate shadow boundary at the aliased location.

An overview of the entire RBSM algorithm is illustrated in Figure 2 and is detailed as follows. RBSM takes as input the shadow map and the scene rendered from the camera viewpoint, with the aliased hard shadows estimated by the shadow test (Figure 2-(a)). After the evaluation of the spatial coherency between neighbours in the shadow map (Figure 2-(a)), the algorithm proceeds by detecting the directions of where aliasing artifacts are located (Figure 2-(b)). In RBSM, these directions (green arrows in Figure 2-(b)) are represented as discontinuities in the shadowing process. On the basis of the discontinuity representation, the next step of RBSM consists in the traversal of the lit side of the shadow boundary (Figure 2-(c)). This traversal is performed with the goal of computing not only the size of the aliased boundary where the fragment is located, but also the relative position of the fragment in this aliased boundary. After the traversal is ended in all directions, RBSM computes the size of the aliased boundary and the distance of each fragment to the ends of the shadow boundary. This distance is normalized to the origin of the local coordinate system of the aliased boundary, as shown in Figure 2-(d). Finally, a linear comparison between vertical and horizontal normalized distances is computed to determine whether a fragment must be revectorized (put in shadow) by the algorithm (Figure 2-(e)).

To formalize the process that happens before the RBSM, let us call a surface point visible in the camera view as $\mathbf{p}$. Also, let us define the distance of $\mathbf{p}$ to the point light source as $\mathbf{p}_z$. Let us refer to each shadow map texel as $\mathbf{t}_{x,y}$, where x and y are the horizontal and vertical texture coordinates of $\mathbf{t}$. Let us also define $z(\mathbf{t}_{x,y})$ as a function that retrieves the distance value of the light blocker of $\mathbf{p}$ stored in the corresponding shadow map texel $\mathbf{t}_{x,y}$. Then, the shadow test proposed in the traditional shadow mapping technique can be formulated as $s(\mathbf{p}_z, z(\mathbf{t}_{x,y}))$

$$s(\mathbf{p}_z, z(\mathbf{t}_{x,y})) = \begin{cases} 0 & \text{if } \mathbf{p}_z > z(\mathbf{t}_{x,y}), \\ 1 & \text{otherwise.} \end{cases} \quad (1)$$

To minimize the aliasing problem caused by the finite resolution of the shadow map, RBSM first evaluates the shadow test for the 4-connected neighbourhood of each lit fragment projected on the shadow map, as follows

$$\mathbf{N} = \left[s(z(\mathbf{t}_{x-o_x,y})), s(z(\mathbf{t}_{x+o_x,y})), s(z(\mathbf{t}_{x,y+o_y})), s(z(\mathbf{t}_{x,y-o_y})) \right], \quad (2)$$

where o_x and o_y are shadow map offset values (the inverse of the shadow map width and height)[1].

Given the neighbourhood evaluation, the next step of RBSM determines the directions where the aliasing artifacts, or shadow discontinuities, are located (Figure 2-(b)). In this context, discontinuity is simply defined as the absolute difference of neighbour shadow tests, which can be stored in an integer array

$$\mathbf{d} = \left[|\mathbf{N}_x - s|, |\mathbf{N}_y - s|, |\mathbf{N}_z - s|, |\mathbf{N}_w - s| \right]. \quad (3)$$

As calculated in Equation (3), $\mathbf{d}$ is a four-dimensional vector that stores the coherency of neighbour shadow tests. Hence, rather than being used to detect the directions of the shadow boundary in a four-dimensional space, $\mathbf{d}$ is simply used to detect whether a shadow boundary exists for each one of the four possible 2D directions (i.e., $\mathbf{d}_x$ for the left direction, $\mathbf{d}_y$ for the right direction, $\mathbf{d}_y$ for the top direction, and $\mathbf{d}_z$ for the bottom direction). For instance, $\mathbf{d}_x = 1$ indicates that the shadow boundary is located at the left side of a fragment. $\mathbf{d}_x = 0$ indicates the opposite case, where the shadow boundary is not located at the left side of a fragment.

The following step of RBSM consists in the shadow boundary traversal. For every lit fragment inside the shadow boundary, the algorithm performs a search to find the ends of the shadow boundary in all the four directions of the 2D space (Figure 2-(c)). For each shadow map neighbour being accessed in a specific direction, RBSM computes the shadow test in Equation (1) and the discontinuity information in Equation (3) to determine whether the neighbour fragment is in shadow or is lit, but does not have any discontinuity direction. Both cases characterize that the neighbour fragment being traversed is out

[1] $\mathbf{p}_z$ was omitted from Equation (2) because $\mathbf{p}_z$ has the same value in the four shadow tests.

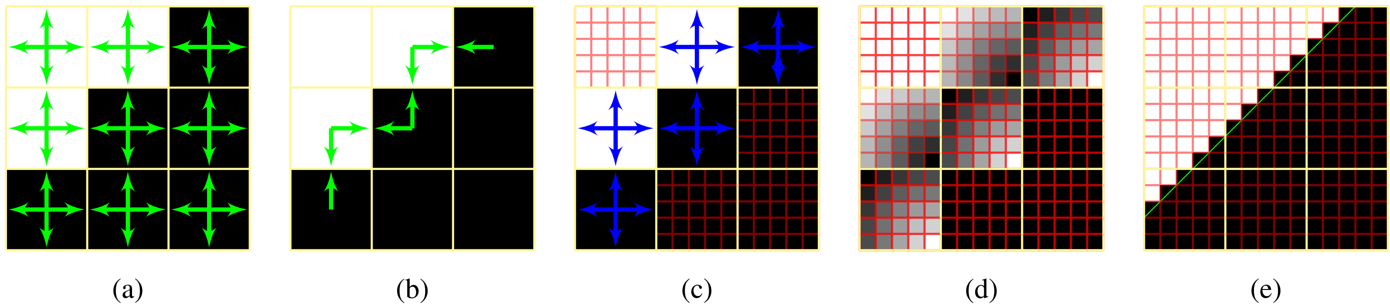


Fig. 3. An overview of the improved visibility function for RBSM. First, we compute the shadow test for every fragment visible in the camera view. Then, for each fragment projected on the shadow map, we evaluate the shadow test of neighbour shadow map texels (a) and detect the directions where the aliasing is located (b), regardless of whether the fragment is located in the inner- or the outer-side of the shadow boundary. Then, for the fragments located in the aliased boundary, we perform the traversal over the shadow boundary to compute the size of the shadow boundary (c) and the normalized relative distance of the fragments to the origin of the local aliasing (d). Taking advantage of the normalized relative distance computed for both sides of the shadow boundary, we are able to compute a dilated version of the revectorized shadow, with less overestimation artifacts than the original approach (e).

of the shadow boundary, indicating that the traversal must be ended for that particular direction. While the rotation of the light source influences the visual aspect of the shadow aliasing seen from the camera viewpoint, the generated shadow map is still aligned with the light source coordinate system. Therefore, the traversal provided by RBSM over X and Y axis of the shadow map works well regardless of the orientation of the light source.

Let us define the normalized distance of the fragment to the shadow edge as α, where α_x and α_y store the normalized distance to the horizontal and vertical ends of the shadow boundary, respectively. Given the shadow boundary orientation shown in Figure 2-(d), where the origin of the local coordinate system is located in the corner of the aliasing, the RBSM visibility function $v(\alpha_x, \alpha_y)$ to produce a more anti-aliased hard shadow (Figure 2-(e)) can be computed as follows

$$v(\alpha_x, \alpha_y) = \begin{cases} 0 & \text{if } 1.0 - \alpha_x > \alpha_y, \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

While this way of revectorizing shadow boundaries is able to suppress aliasing artifacts with reasonable accuracy, the algorithm uses a conservative approach to perform the shadow anti-aliasing because it operates only on the lit side of the shadow boundary, achieving a small overhead compared to shadow mapping, but introducing the overestimation of the shadow boundary. As shown in Figure 5-(b), details of the shadow may be lost mainly for parts of the shadow that are too near to each other, since the algorithm will tend to merge, or at least approximate, these originally disconnected regions. To solve this problem, we propose an adaptation of the RBSM pipeline to reduce the overestimation of the hard shadow anti-aliasing.

3.2. *Improved Shadow Anti-Aliasing*

The main problem of RBSM is that, by working only over the lit side of the shadow boundary, the technique is able to recover an approximate shadow boundary, but suffers from overestimation artifacts because the real, accurate shadow boundary is not located only in the external part of shadows produced by shadow mapping. By analyzing both sides (lit and shadowed), we can reduce the overestimation. In this sense, to improve the anti-aliasing provided by RBSM, we propose an extension of its pipeline to use both lit and shadowed sides of the shadow boundary for hard shadow revectorization.

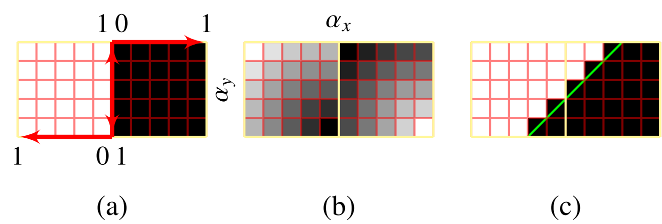
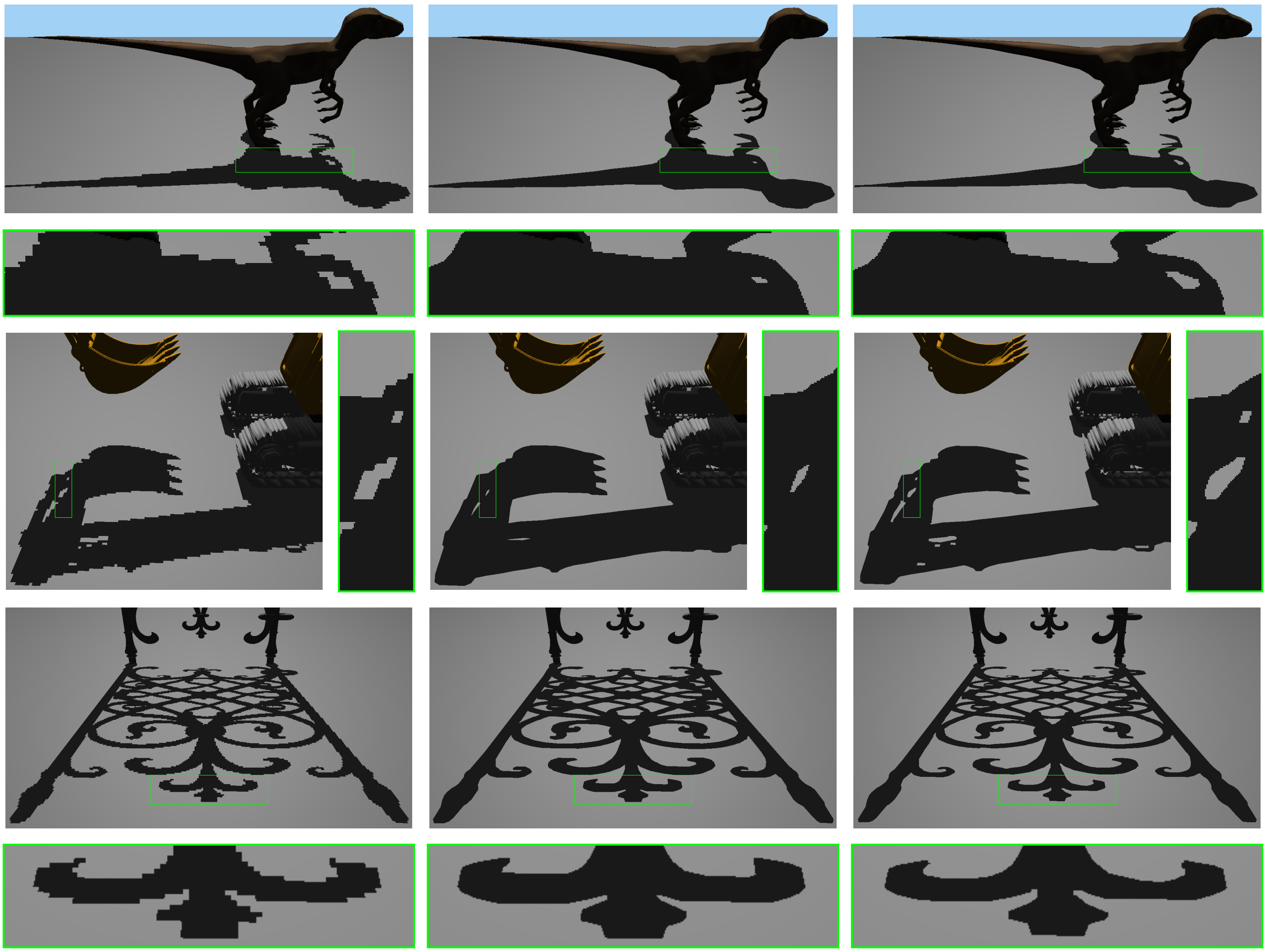


Fig. 4. The local coordinate system shown in (a) is used as reference for the calculation of the normalized distance (α) of the fragment to the origin of the aliased shadow boundary (b). On the basis of this value, we can fit the revectorization line (green line in (c)) that will define the visibility condition of each fragment.

An overview of the new RBSM pipeline is shown in Figure 3. As can be seen in Figure 3-(a), after the shadow map generation, both shadow test and neighbourhood evaluation given in Equations (1) and (2) are computed for all lit and now also shadowed fragments visible in the camera view. Then, on the basis of the neighbourhood evaluation, the algorithm detects all the fragments located in the shadow boundary using the same discontinuity test and computes the discontinuity directions in Equation (3) for both sides of the shadow boundary (Figure 3-(b)). For each fragment in the aliased shadow boundary, a traversal is performed for both sides of the shadow boundary (Figure 3-(c)) to estimate the aliasing size and the normalized relative distance of each fragment to the shadow boundary (Figure 3-(d)). Then, a new visibility function calculation is used to determine the new location of the revectorized shadow boundary (Figure 3-(e)).

During the shadow boundary traversal, the shadow test in Equation (1) is computed for every neighbour shadow map texel being accessed. To detect the end of the shadow boundary, we check if the neighbour fragment has a different visibility condition than the one estimated by the initial fragment of the traversal. In this sense, for lit fragments, the shadow boundary ends in a shadowed fragment. On the counterpart, for shadowed fragments, the shadow boundary ends in a lit fragment. If



(a) Shadow Mapping (b) Original RBSM (c) Improved RBSM

Fig. 5. The original visibility function of RBSM suppresses aliasing artifacts generated by shadow mapping (a), but overestimating the shadow (b). The proposed new visibility function is able to minimize aliasing artifacts with less overestimation artifacts (c). Images were generated for the Raptor (top), Excavator (middle) and Fence (bottom) models using a 512^2, 1024^2 and 2048^2 shadow map resolutions.

the visibility condition between neighbours is the same, we still compute the discontinuity directions in Equation (3) and check whether neighbour fragments share at least one discontinuity direction. If that is not the case, the traversal has stepped out of the lit/shadowed side of the aliased shadow boundary.

Once the traversal has ended, we proceed with the computation of the normalized distance of each fragment to the origin of the aliased shadow boundary (Figure 3-(d)). Depending on whether the fragment is located inside or outside the shadowed part of the boundary, the origin of this local coordinate system is changed. Nevertheless, the origin is still located at the corner of the aliasing.

Given the orientation shown in Figure 4-(a), the normalized distances α_x and α_y (Figure 4-(b)), and the shadow test s in Equation (1), we can first analyze the proposed improved visibility function $v(s, \alpha_x, \alpha_y)$ for the separate cases of when the fragment is lit ($s = 1$) or is in shadow ($s = 0$):

$$v(0, \alpha_x, \alpha_y) = \begin{cases} 0 & \text{if } 1.0 - \alpha_x - \alpha_y < 0.5, \\ 1 & \text{otherwise,} \end{cases} \quad (5)$$

$$v(1, \alpha_x, \alpha_y) = \begin{cases} 0 & \text{if } \alpha_x + \alpha_y < 0.5, \\ 1 & \text{otherwise.} \end{cases} \quad (6)$$

As shown in Figure 4-(c), for the originally shadowed part of the shadow boundary (right yellow texel of Figure 4-(a)), whenever the distance α is greater or equal than 0.5, the shadowed part is changed to be lit in Equation (5). Likewise, for the originally lit part of the shadow boundary (left yellow texel of Figure 4-(a)), whenever the distance α is lower than 0.5, the lit part is put in shadow, as shown in Equation (6).

As can be seen in Equations (5, 6), the visibility functions are in some way complementary to each other. So, we can redefine the visibility function as

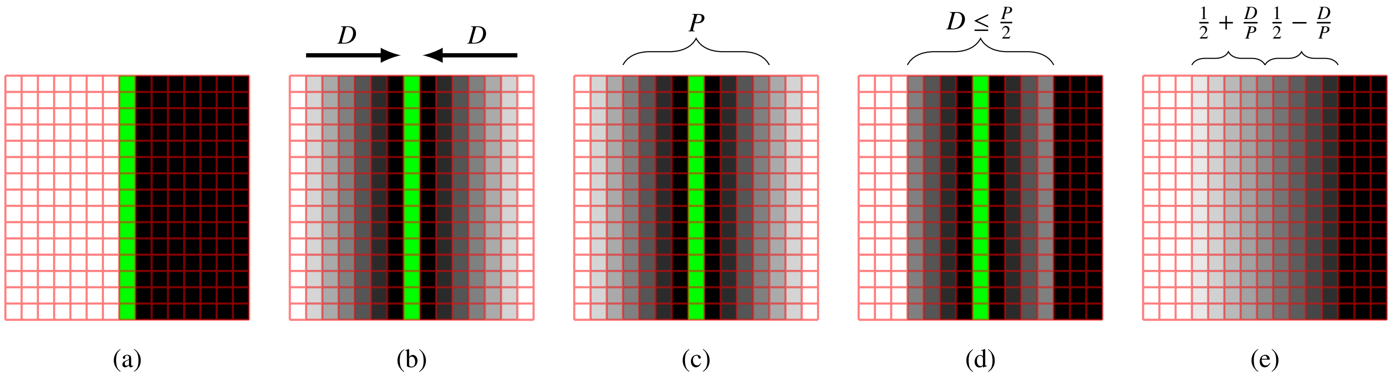


Fig. 6. An overview of the EDTSM. First, the RBSM is used (a) to generate anti-aliased shadow boundaries (green rectangles) in the camera view (red grid). Then, for every fragment in the screen space, the world-space position is retrieved from the G-buffer, and the world-space distance D to closest fragment located in the shadow boundary is computed (b). Given a user-defined penumbra size P (c), the algorithm restricts the penumbra computation for fragments located in the penumbra region (d). Finally, the EDT previously computed is normalized to simulate the smooth transition between lit and umbra regions that characterize the penumbra effect (e).

$$v(s, \alpha_x, \alpha_y) = \begin{cases} 0 & \text{if } (1-s) + (2s-1)(\alpha_x + \alpha_y) < 0.5 \\ 1 & \text{otherwise.} \end{cases} \quad (7)$$

In Figure 5, we show a comparison between the hard shadows produced with shadow mapping (Figure 5-(a)), the original RBSM visibility function (Figure 5-(b)) and the proposed improved visibility function (Figure 5-(c)). By using the visibility function of Equation (7), we can properly revectorize the shadow boundary, keeping the original details of the shadow captured by shadow mapping, and performing the anti-aliasing with less overestimation artifacts than the original RBSM visibility function (Figure 5-(c)).

3.3. Euclidean Distance Transform Penumbra Simulation

Let us call *seed* a fragment that lies in the hard shadow boundary (green rectangles in Figure 6). It will be used as a basis for the EDT computation. Even if it is located in a thin aliased shadow, a seed fragment can be easily located in the screen space of the camera view by the application of a 3×3 rectangular filter over the shadows produced with the improved RBSM. In this case, a fragment is a seed if the hard shadow intensity of the fragment differs from the hard shadow intensity of one of its neighbours located in the 8-connected neighbourhood of the fragment in the screen space (Figure 6-(a)).

Once the seed fragments have been detected in the image, the EDT can be computed. So, for each non-seed fragment, the world-space Euclidean distance D of the fragment to the nearest seed located in the shadow boundary is computed (Figure 6-(b)), D being a world-space distance computed on the basis of the world-space position retrieved from a G-buffer [33] previously computed. Just by applying the EDT in the world space, the user does not have control over the desired penumbra size. To solve this problem, let us assume P as a user-defined parameter which controls the size of the penumbra that will be simulated. As shown in Figure 6-(c), each half of the penumbra size belongs to one side of the shadow boundary. Therefore, one can easily detect whether a fragment belongs to the desired penumbra region by checking if the distance of the fragment to the shadow boundary is lower or equal than half of the penumbra size (i.e., $D \leq P/2$) (Figure 6-(d)). For the fragments located outside of the penumbra region, the shadow intensity is given by the shadow test (umbra and lit regions in Figure 6-(d)). Meanwhile, for the fragments in the penumbra region, we keep the result of the EDT as shadow intensity.

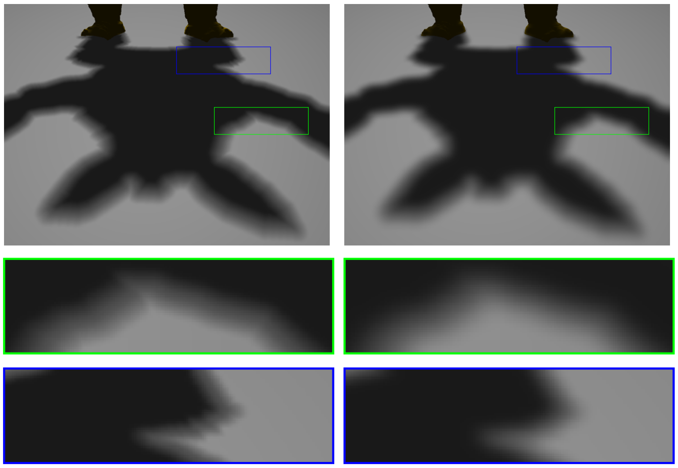
As can be seen in Figure 6-(d), EDT does not resemble a penumbra mostly because it is not normalized. The transition between umbra and lit regions in the desired penumbra is not smooth as it should be to characterize a penumbra. To solve this problem, the Euclidean distance is normalized to the closed unit interval $[0, 1]$, assuming that umbra and lit fragments have intensities 0 and 1, respectively. Hence, the final intensity I of the fragments located in the penumbra region is

$$I = \begin{cases} \frac{1}{2} - \frac{D}{P} & \text{if the fragment was in shadow,} \\ \frac{1}{2} + \frac{D}{P} & \text{otherwise.} \end{cases} \quad (8)$$

As shown in Equation (8), the final intensity of each fragment depends on the previous visibility condition given by RBSM. For instance, knowing that the maximum distance of each fragment belonging to the penumbra region to the nearest seed is $P/2$ (Figure 6-(d)), if the fragment was in shadow, as computed by RBSM, the new penumbra intensity of the fragment must lie in the interval $[0, 0.5]$, because 0 is the intensity of the fragments located in shadow and 0.5 is the intensity of the fragments located in the middle of the penumbra region. Accordingly, lit fragments, as computed by RBSM, must have their penumbra intensities lying in the interval $[0.5, 1]$. By the use of Equation (8), EDTSM is able to satisfy these constraints and simulate the penumbra effect (Figure 6-(e)).

3.4. Euclidean Distance Transform Shadow Filtering

EDT is well known to generate skeletons along gradient discontinuities [34]. This property of EDT is desirable in several applications, such as integer medial axis estimation [35]. However, these skeletons generated by EDT, when visualized inside a penumbra, constitute an artifact because a penumbra does not



(a) (b)

Fig. 7. After the EDT computation, skeleton artifacts may arise along gradient discontinuities (green closeup of (a)). Also, aliasing artifacts may still remain even after the shadow revectorization (blue closeup of (a)). By applying a simple mean filter, those artifacts can be suppressed (b). Images were generated for the Armadillo model using a 512^2 shadow map resolution.

have skeletons along its boundary (green closeup of Figure 7-(a)). Moreover, RBSM is able to minimize aliasing artifacts generated by shadow mapping, but is not able to remove all of them (blue closeup of Figure 7-(a)). To minimize both skeleton and aliasing artifacts simultaneously, a simple screen-space separable mean filter is applied over the shadow boundary (Figure 7-(b)). The mean filtering was chosen to solve these problems because of its simplicity, low processing time, separability and effectiveness to suppress the skeleton artifacts even for low-order filter sizes.

The EDT algorithm is performed in screen space, taking as input the image of the shadowed scene rendered from the camera viewpoint. In this sense, special care must be taken to make this process edge aware and viewpoint invariant. Edge awareness is important because different objects cannot influence on the penumbra computation of each other. Viewpoint invariance is desirable because the penumbra size must be kept constant, regardless of the distance of the viewer to the shadowed region. EDTSM solves both problems by the use of the depth value and world-space position stored in the G-buffer [33]. Depth information is used to detect edges, which separate different objects in the scene. In this sense, for instance, a fragment is only considered to be in penumbra if the depth difference between the fragment and its nearest seed is below a user-defined threshold (empirically we have set this depth threshold as 2.5×10^{-3}). Also, only neighbours with similar depth difference are taken into account for mean filtering. In counterpart, to make the EDT viewpoint invariant, world-space position is used to compute the Euclidean distance values D in the EDT. Inspired by screen-space soft shadow algorithms, the viewpoint invariance of the mean filtering is solved by estimating the mean filter size w_{filter}^{screen} that varies according to the distance of the camera to the scene. Here, w_{filter}^{screen} is measured as [36]

$$w_{filter}^{screen} = \frac{w_{filter} z_{screen}}{\mathbf{p}_{z_{eye}}}, \tag{9}$$

$$z_{screen} = \frac{1}{2\tan\frac{fov_y}{2}}, \tag{10}$$

where w_{filter} is the mean filter size defined by the user, $\mathbf{p}_{z_{eye}}$ is the distance of the fragment $\mathbf{p}$ to the center of the camera and fov_y specifies the vertical field of view angle.

4. Results and Discussion

In this section, we compare the original EDTSM [15] and the proposed improved EDTSM (hereafter named as EDTSM* only for clarity) with traditional shadow mapping techniques, such as shadow mapping itself [5], PCF [9], VSM [11], and EVSM [28], as well as state-of-the-art penumbra simulation techniques, such as MSM [14] and RPCF [10]. With exception of shadow mapping, all these techniques were chosen for performance and visual quality evaluation because they produce fixed-size penumbra, lying in the scope of this paper. Therefore, techniques that simulate variable-size penumbra (i.e., soft shadows) on the basis of point or area light sources are not evaluated in this section.

We have tested different penumbra simulation techniques in three distinct scenarios. Figure 8 shows a model with fine detailed structures along its boundary. Figure 9 shows shadows cast on a non-planar model. Figure 10 shows a scenario with several light blocker and shadow receiver objects positioned over each other. In the supplementary video, we show additional visual results of the proposed EDTSM* algorithm, including temporal consistency.

4.1. *Experimental Environment*

A computer equipped with an Intel Core^{TM} i7-3770K CPU (3.50 GHz), 8GB RAM, and an NVIDIA GeForce GTX Titan X graphics card, was used to run the experimental tests. OpenGL [37] and GLSL [38] languages were used to implement the EDTSM algorithm. For EDT computation, we have used the open-source implementation of the parallel banding algorithm (PBA) [16] implemented in CUDA [39]. Although many other works have attempted to compute EDT efficiently [40, 41, 42], in our tests, PBA delivered the fastest and most accurate EDT computation. CUDA/OpenGL interoperability was used to optimize resource management and processing time. A filter of order 15×15 is used to suppress skeleton and banding artifacts. For RPCF, we have used a filter of order 7×7, as suggested in [10].

4.2. *Rendering Quality*

In the **blue** closeups of Figures 8 and 9, we show whether the different shadowing techniques are able to suppress aliasing artifacts. Without taking advantage of any anti-aliasing strategy, the traditional shadow mapping generates aliasing artifacts along the shadow boundary (Figures 8-(a), 9-(a) and 10-(a)). For small penumbra sizes, the blur provided by PCF is insufficient to suppress aliasing artifacts (Figures 8-(b) and 9-(b)).

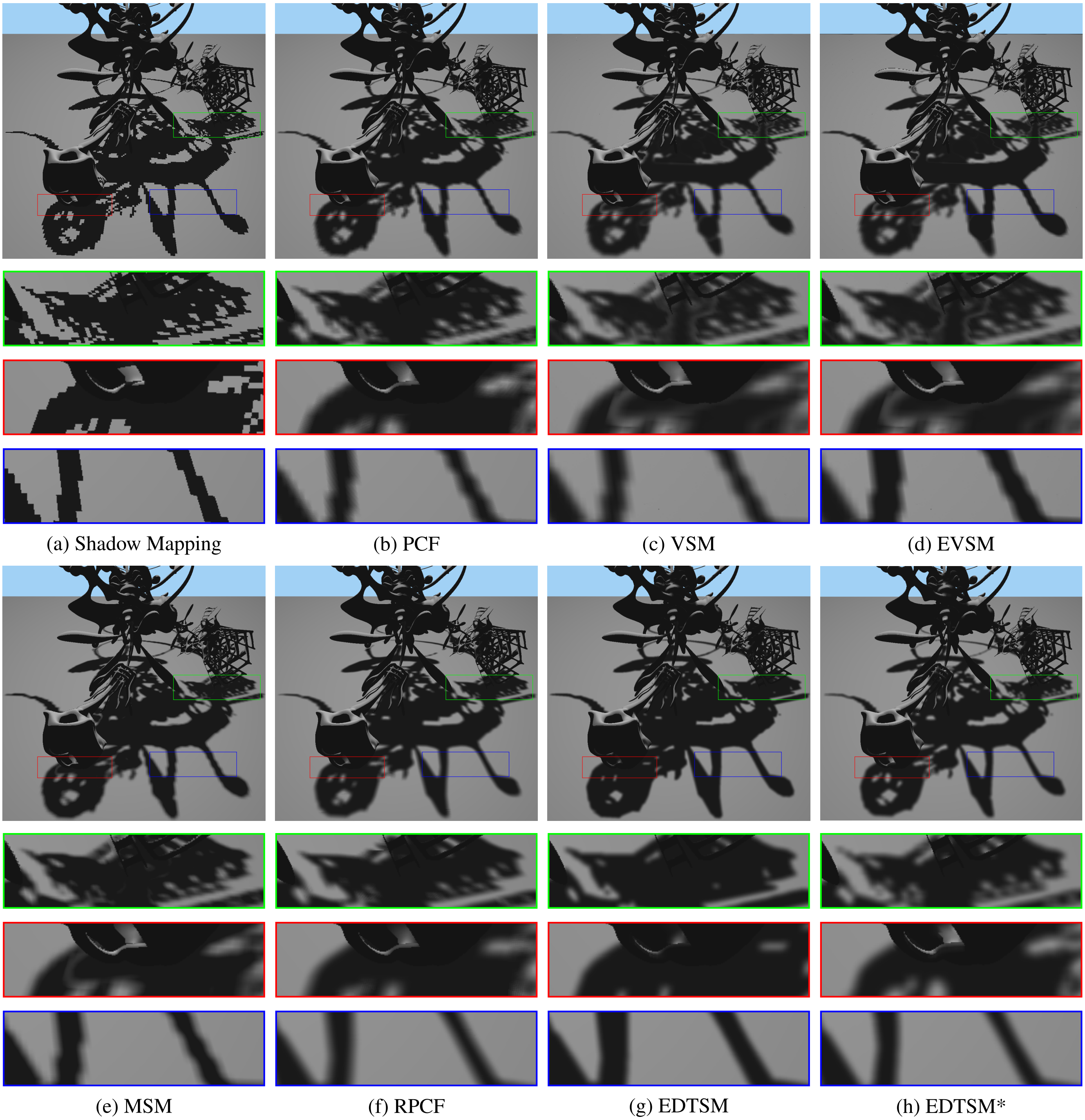


Fig. 8. Fixed-size penumbra produced by different techniques. Each closeup shows whether the technique handles overestimation (green), light leaking (red) and aliasing (blue) artifacts. Images were generated for the YeahRight model using a 1024^2 shadow map resolution.

The same effect is visible for shadow map filtering techniques (Figures 8-(c, d, e) and 9-(c, d, e)), which simulate penumbra with blurred jagged boundaries along the shadow. Techniques that use shadow revectorization as basis for penumbra simulation (RPCF, EDTSM and EDTSM*) are able to minimize this artifact efficiently (Figures 8-(f, g, h) and 9-(f, g, h)).

In the **red** closeups of Figures 8, 9 and 10, we show whether a technique generates light leaking artifacts inside the shadow. Taking as reference the shadows generated by shadow mapping (Figures 8-(a), 9-(a) and 10-(a)), all shadow map filtering techniques are prone to light leaking artifacts. In this sense, VSM (Figures 8-(c), 9-(c) and 10-(c)) is more susceptible to light leaking than EVSM (Figures 8-(d), 9-(d) and 10-(d)). In terms of visual quality, MSM is better than both VSM and EVSM, greatly reducing the light leaking artifacts (the effectiveness of MSM is mainly visible in Figure 9-(e)). The techniques that

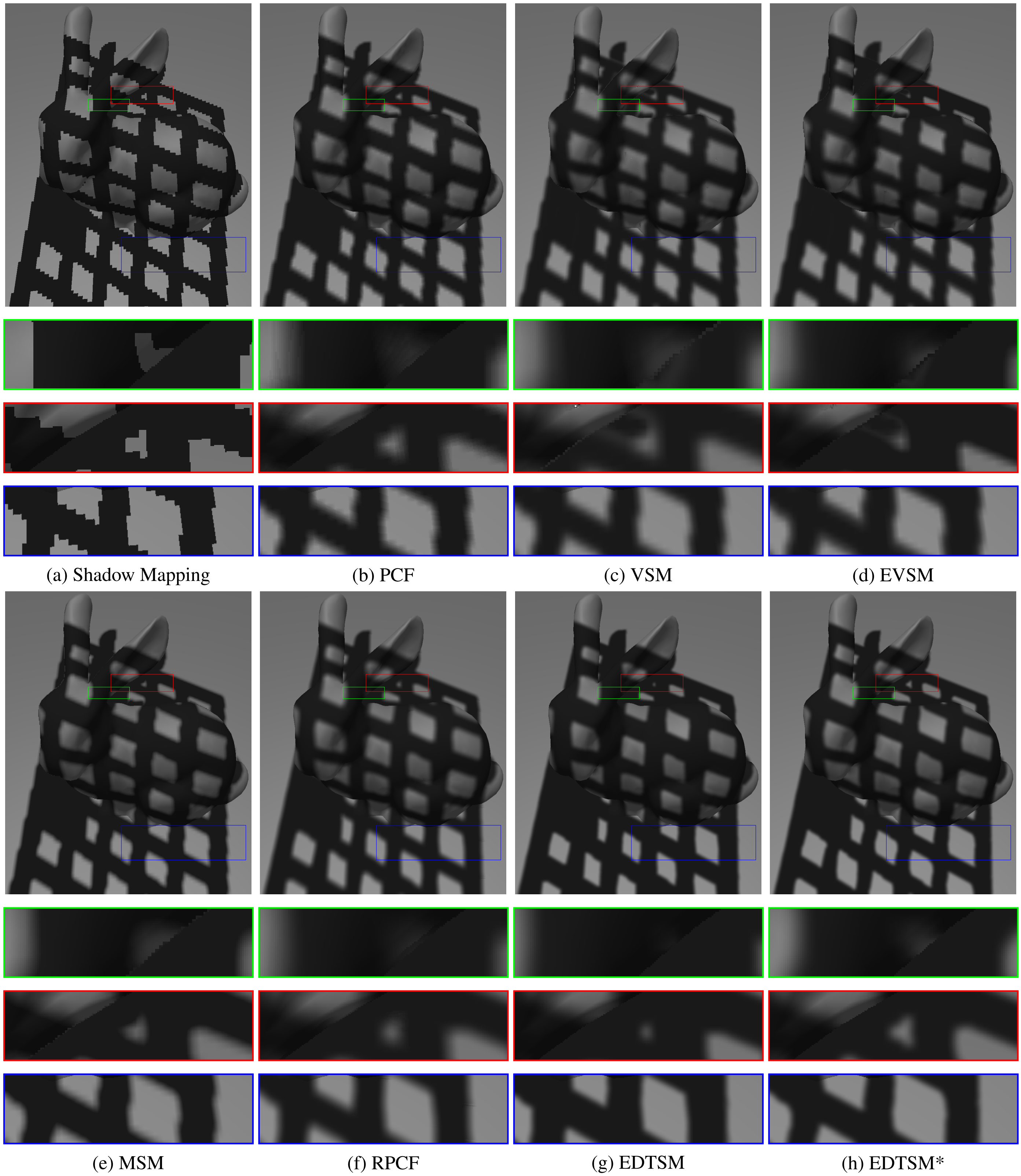


Fig. 9. Fixed-size penumbra produced by different techniques. Each closeup shows whether the technique handles overestimation (green), light leaking (red) and aliasing (blue) artifacts. Images were generated for the Bunny model using a 1024^2 shadow map resolution.

filter shadows to simulate penumbra (PCF, RPCF, EDTSM and EDTSM*) are not prone to light leaking artifacts (Figures 8-(b, f, g, h), 9-(b, f, g, h) and 10-(b, f, g, h)).

In the **green** closeups of Figures 8, 9 and 10, we zoom in parts of shadows that may be suppressed by the shadowing technique, causing the overestimation artifact. PCF filters the

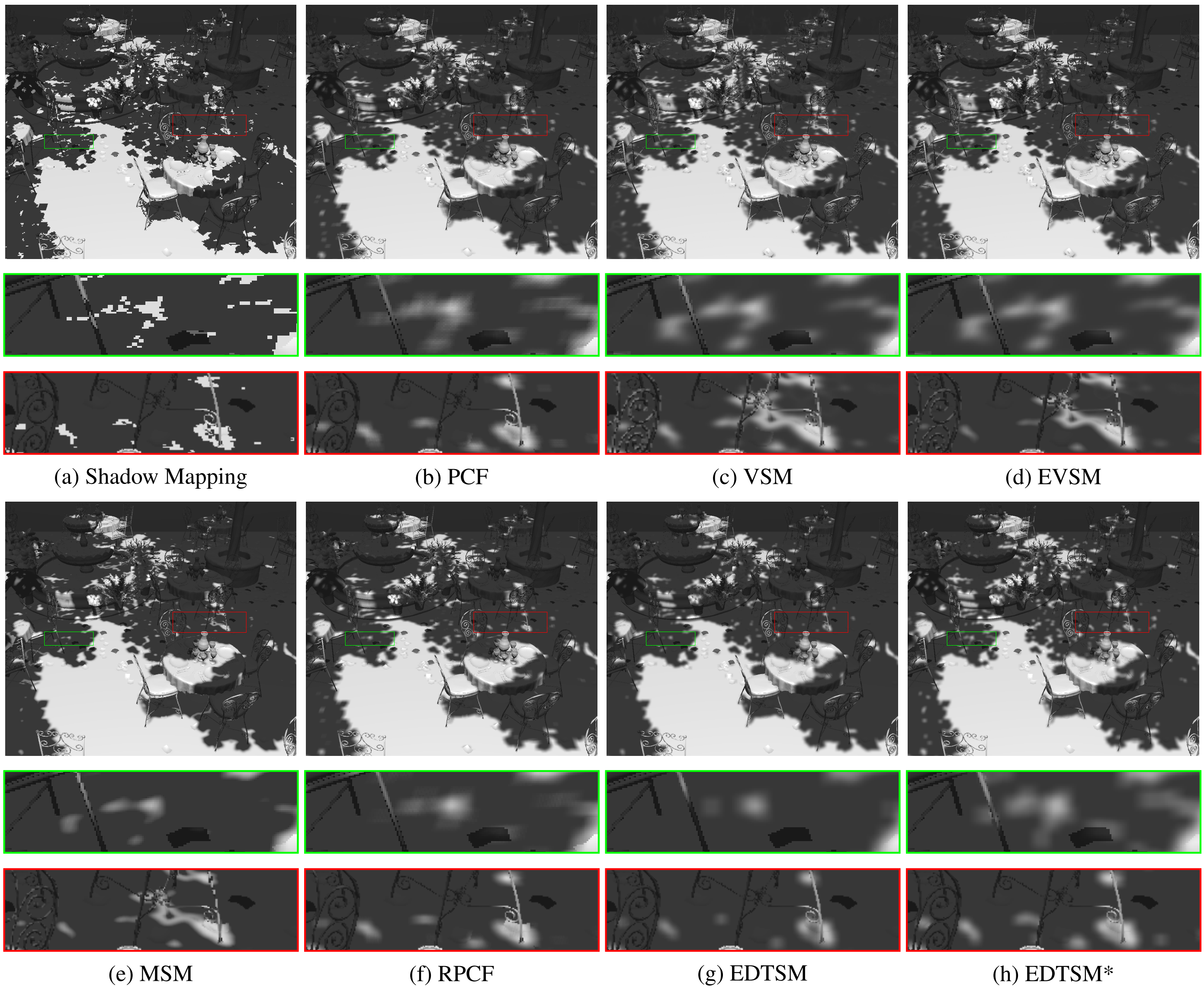


Fig. 10. Fixed-size penumbra produced by different techniques. Each closeup shows whether the technique handles overestimation (green) or light leaking (red) artifacts. Images were generated for the SanMiguel model using a 1024^2 shadow map resolution.

shadow test and simulates penumbra without overestimation artifacts (Figures 8-(b), 9-(b) and 10-(b)). Rather than suffering from shadow overestimation, VSM introduces light leaking artifacts (Figures 8-(c), 9-(c) and 10-(c)). EVSM tries to minimize the light leaking artifacts of VSM without introducing the shadow overestimation (Figures 8-(d), 9-(d) and 10-(d)). On the other hand, MSM minimizes the light leaking artifacts of VSM and EVSM at the cost of the slight overestimation of shadows (this effect is mainly visible in Figure 10-(e)). RPCF is based on a filtering variant of RBSM [10] and suffers from overestimation artifacts (Figures 8-(f), 9-(f) and 10-(f)). These artifacts are further worsened in EDTSM (Figures 8-(g), 9-(g) and 10-(g)). Compared to the other techniques, the penumbra simulated by EDTSM is the one with less details along the shadow because of the shadow overestimation. By reformulating the RBSM visibility function, we could address this problem efficiently in EDTSM*, recovering details that were missed by EDTSM and producing shadows with less artifacts than related work (Figures 8-(h), 9-(h) and 10-(h)).

The proposed EDTSM* supports shadow rendering for planar (Figure 8-(h)) and non-planar receivers (Figures 9-(h)). Moreover, EDTSM* supports penumbra simulation not only for simple scenarios, but also for more complex, game-like scenarios, such as the one shown in Figure 10-(h), where several light blocker and shadow receiver objects with fine detailed structures (e.g., trees) are located in the same scene. Also, EDTSM* supports penumbra simulation on noisy surfaces with high-frequency details, as shown in Figure 12. In this figure, we reinforce that EDTSM* is able to separate penumbra simulation from self-shadowing, while properly handling the noisy depth differences distributed over the surface. Hence, the gamut of scenes shown in this section reveals that EDTSM* is able to generate fixed-size penumbra, with less aliasing, light leaking and shadow overestimation artifacts than related work.

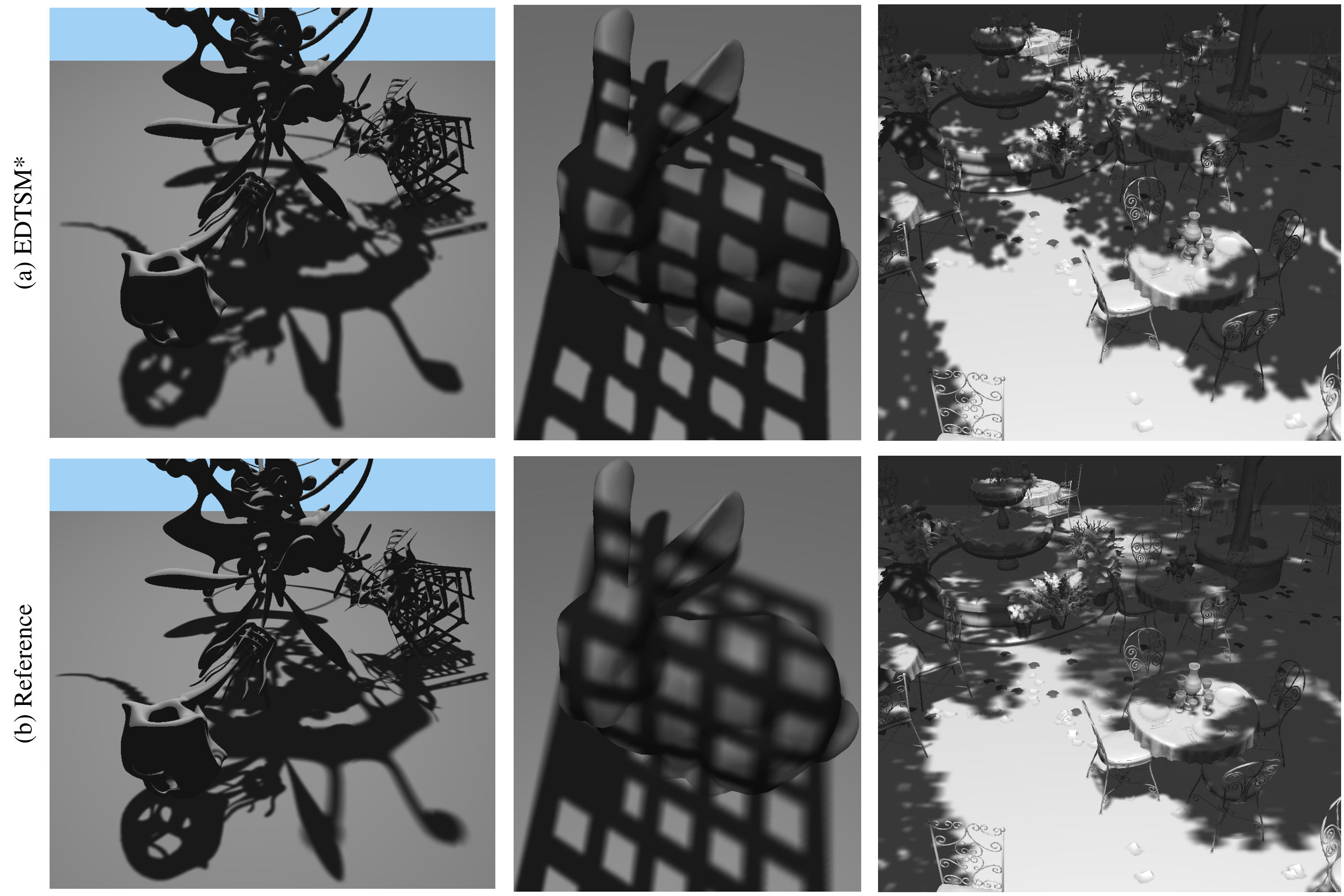


Fig. 11. A comparison between shadows generated by EDTSM* (top) and the ground-truth technique (bottom) for three scenes shown in this paper. Ground-truth images were computed using the average of 1024 samples from an area light source.

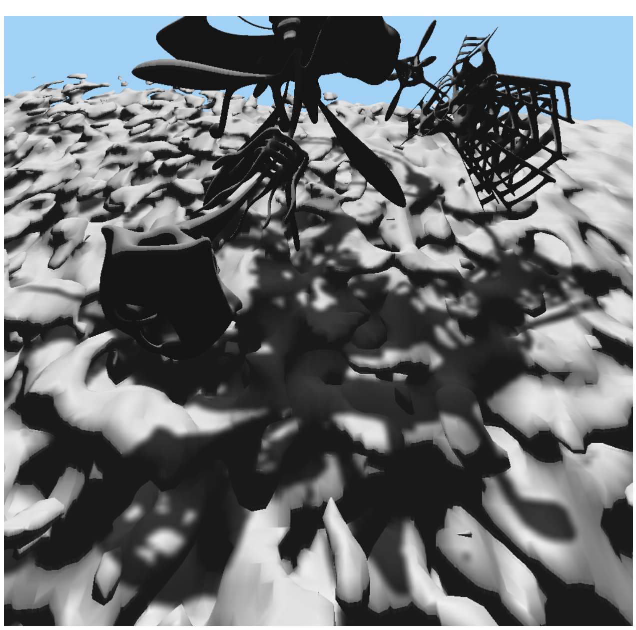


Fig. 12. Fixed-size penumbra simulation on a noisy surface with laterally increasing frequency. Image was generated for the YeahRight model using a 1024^2 shadow map resolution.

EDTSM* lies in the category of shadow mapping techniques that simulate fixed-size penumbra on the basis of a point light source to achieve real-time performance (Figure 11-(a)). Unfortunately, shadows produced by fixed penumbra shadowing techniques do not capture the realism of ground-truth soft shadows (Figure 11-(b)), mainly because real-world penumbra has a variable size that varies according to the distance of each shadow receiver fragment to both light blocker fragments and the area light source.

4.3. Processing Time

As shown in Tables 1 and 2, shadow mapping is the fastest shadowing technique, but is not able to simulate penumbra (Figures 8-(a), 9-(a) and 10-(a)). Shadow map filtering is an efficient way to simulate penumbra, being slightly slower than shadow mapping, and relatively scalable with respect to the shadow map (Table 1), viewport (Table 2) and kernel resolutions (Table 3). However, all this efficiency comes at the price of light leaking artifacts generation inside shadows (Figures 8-(c, d, e), 9-(c, d, e) and 10-(c, d, e)). PCF is more scalable to the shadow map resolution than the shadow map filtering techniques (Table 1), but is one of the slowest techniques for high-order filter sizes (Table 3) and is prone to aliasing artifacts along the shadow boundary (Figures 8-(b) and 9-(b)). RPCF is able to suppress

Table 1. Processing time for several hard shadow filtering techniques and different scenes rendered at an output 1280×720 resolution. Measurements include varying shadow map resolution. SM - Shadow Mapping. PF - Pre-filtering techniques (namely VSM, EVSM and MSM).

		Shadow Map Resolution (ms)			
Scene	Method	512^2	1024^2	2048^2	4096^2
Figure 8	SM	9.3	9.4	9.5	9.9
	PF	10.5	10.6	10.8	12.0
	PCF	11.3	11.4	11.4	11.7
	EDTSM	12.8	12.9	13.0	13.7
	EDTSM*	12.9	13.0	13.2	13.8
	RPCF	26.2	27.3	27.8	30.0
Figure 9	SM	0.7	0.8	0.9	1.3
	PF	1.9	2.1	2.4	3.6
	PCF	2.9	3.0	3.1	3.5
	EDTSM	4.4	4.5	4.7	5.6
	EDTSM*	4.5	4.6	4.8	5.7
	RPCF	10.4	11.0	11.5	12.0
Figure 10	SM	126.2	127.2	127.8	131.1
	PF	126.9	128.2	129.8	133.8
	PCF	126.7	129.0	130.3	133.5
	EDTSM	128.8	130.2	132.2	135.8
	EDTSM*	129.5	130.9	132.8	136.3
	RPCF	138.8	141.2	145.7	148.1

Table 2. Processing time for several hard shadow filtering techniques and different scenes rendered at a 1024^2 shadow map resolution. Measurements include varying output image resolution. SM - Shadow Mapping. PF - Pre-filtering techniques (namely VSM, EVSM and MSM).

		Output Resolution (ms)		
Scene	Method	$480p$	$720p$	$1080p$
Figure 8	SM	8.8	9.4	9.7
	PF	9.9	10.6	11.3
	PCF	9.8	11.4	11.9
	EDTSM	10.7	12.9	14.7
	EDTSM*	10.8	13.0	14.9
	RPCF	16.4	27.3	30.3
Figure 9	SM	0.4	0.8	1.1
	PF	0.9	2.1	3.1
	PCF	0.9	3.0	4.0
	EDTSM	2.3	4.5	6.9
	EDTSM*	2.3	4.6	6.9
	RPCF	4.3	11.0	12.5
Figure 10	SM	127.1	127.2	127.5
	PF	127.8	128.2	128.8
	PCF	127.8	129.0	129.5
	EDTSM	128.3	130.2	132.6
	EDTSM*	129.3	130.9	133.5
	RPCF	132.9	141.2	143.7

Table 3. Processing time for several hard shadow filtering techniques and different scenes rendered at an output 1280×720 resolution and using a 1024^2 shadow map resolution. Measurements include varying kernel size. SM - Shadow Mapping. PF - Pre-filtering techniques (namely VSM, EVSM and MSM).

		Kernel Size (ms)			
Scene	Method	7^2	15^2	23^2	31^2
Figure 8	SM	9.4	9.4	9.4	9.4
	PF	10.3	10.6	10.8	11.1
	PCF	9.8	11.4	13.5	17.0
	EDTSM	12.3	12.9	13.5	14.2
	EDTSM*	12.4	13.0	13.5	14.3
	RPCF	27.3	77.5	166.6	285.7
Figure 9	SM	0.8	0.8	0.8	0.8
	PF	1.8	2.1	2.4	2.7
	PCF	1.2	3.0	5.6	9.3
	EDTSM	4.0	4.5	5.3	6.1
	EDTSM*	4.1	4.6	5.3	6.1
	RPCF	11.0	65.7	145.3	255.7
Figure 10	SM	127.2	127.2	127.2	127.2
	PF	127.9	128.2	128.4	128.6
	PCF	128.3	129.0	130.2	133.5
	EDTSM	129.6	130.2	130.9	131.9
	EDTSM*	130.3	130.9	131.6	132.6
	RPCF	141.2	200.2	366.3	552.2

aliasing artifacts (Figures 8-(f), 9-(f) and 10-(f)), but is the slowest penumbra simulation technique, regardless of shadow map (Table 1), viewport (Table 2) and kernel resolution (Table 3). EDTSM and EDTSM* are slightly slower than the majority of previous work (Table 1) and are not scalable with respect to the output image resolution (Table 2), because EDT is an image-based operation. Finally, both EDTSM and EDTSM* are more scalable to the filter size than PCF and RPCF techniques, becoming even faster than these two related work for high-order filter sizes (Table 3). While EDTSM suffers from shadow overestimation artifacts, EDTSM* is able to reduce these artifacts, while adding a small overhead of $\approx$1% of the processing time.

In Table 4, we show the processing time obtained for each step of the EDTSM* for varying output resolution. We have not conducted the same analysis for other parameters because a variation in the shadow map resolution (Table 1) affects mainly the shadow map rendering and RBSM computation steps. Meanwhile, the variation of the kernel size (Table 3) affects only the mean filtering step.

From Table 4, we can see that the EDT computation is the bottleneck of EDTSM*. We recall that, to the best of our knowledge, the algorithm that we use to compute the EDT in real-time, the PBA [16], is the fastest algorithm able to compute exact EDT on the GPU. Even in this case, the algorithm still demands more than 2 milliseconds to compute the EDT in $720p$ or higher output resolutions.

5. Conclusion and Future Work

In this paper, we have extended our previous work, published in the proceedings of the Graphics Interface 2017 [15], by improving the anti-aliasing visibility function of the revectorization-based shadow mapping technique, addressing the shadow overestimation problem of the Euclidean distance transform shadow mapping. From a traversal of both sides of the aliased hard shadow boundary, we could define a revectorized boundary with less overestimation than the previous approach. With this improved version of the EDTSM algorithm,

Table 4. Processing time of each individual step of the proposed EDTSM* (including G-buffer and shadow map rendering) for different scenes rendered using a 1024^2 shadow map resolution. Measurements include varying output image resolution.

		Output Resolution (ms)		
Scene	**Step**	$480p$	$720p$	$1080p$
	G-buffer	4.1	4.5	4.7
	Shadow Map	4.3	4.3	4.3
	RBSM	0.2	0.4	0.6
Figure 8	EDT	1.3	2.3	2.9
	Mean Filter	0.7	1.2	2.0
	Shading	0.2	0.3	0.4
	Total	**10.8**	**13.0**	**14.9**
	G-buffer	0.1	0.3	0.4
	Shadow Map	0.2	0.2	0.2
	RBSM	0.1	0.4	0.7
Figure 9	EDT	1.1	2.3	3.1
	Mean Filter	0.7	1.2	2.2
	Shading	0.1	0.2	0.3
	Total	**2.3**	**4.6**	**6.9**
	G-buffer	63.4	63.5	63.6
	Shadow Map	63.0	63.0	63.0
	RBSM	0.6	0.7	1.0
Figure 10	EDT	1.3	2.3	3.3
	Mean Filter	0.7	1.1	2.2
	Shading	0.3	0.3	0.4
	Total	**129.3**	**130.9**	**133.5**

we were able to enhance the visual quality of the fixed-size penumbra simulation, solving most of the artifacts commonly found in the state-of-the-art techniques. With respect to frame rate, the new visibility function slightly increases the processing time to minimize the overestimation artifact. Even so, the proposed EDTSM is more scalable than PCF and RPCF for high-order filter sizes, proving to be a shadowing technique attractive for real-time applications, such as games and augmented reality.

Unfortunately, EDTSM is slightly slower than previous work because the EDT computation is the costly step of EDTSM, even though we make use of the fastest solution proposed so far for EDT computation. Hence, a suggestion for future work is the proposition of a faster, less accurate EDT algorithm to speed up EDTSM. Another option for future work is the extension of EDTSM to compute variable-sized penumbrae in accurate soft shadows.

Acknowledgments

We are thankful to Cao et al. [16] for gently sharing the source code of the parallel banding algorithm. The Fence model shown in this paper is courtesy of Archive3D (user Nike). The Excavator model is courtesy of Free3D (3Dregenerator user). The YeahRight model is courtesy of Keenan Crane. The simplified San Miguel model is courtesy of Morgan McGuire [43]. This research is supported by the scholarship program of Coordenação de Aperfeiçoamento de Pessoal do Nível Superior (CAPES) and by NVIDIA Corporation, who provided the hardware used to evaluate this work, through the GPU Education Center program.

References

[1] Wanger, LC, Ferwerda, JA, Greenberg, DP. Perceiving Spatial Relationships in Computer-Generated Images. IEEE Comput Graph Appl 1992;12(3):44—58.

[2] Hecher, M, Bernhard, M, Mattausch, O, Scherzer, D, Wimmer, M. A Comparative Perceptual Study of Soft-Shadow Algorithms. ACM Trans Appl Percept 2014;11(2):5:1–5:21.

[3] Immel, DS, Cohen, MF, Greenberg, DP. A Radiosity Method for Non-diffuse Environments. In: Proceedings of the ACM SIGGRAPH. 1986, p. 133–142.

[4] Kajiya, JT. The Rendering Equation. In: Proceedings of the ACM SIGGRAPH. 1986, p. 143–150.

[5] Williams, L. Casting Curved Shadows on Curved Surfaces. In: Proceedings of the ACM SIGGRAPH. 1978, p. 270–274.

[6] Williams, L. Pyramidal parametrics. In: Proceedings of the ACM SIGGRAPH. 1983, p. 1–11.

[7] Heckbert, PS. Fundamentals of texture mapping and image warping. Tech. Rep.; 1989.

[8] Bruneton, E, Neyret, F. A Survey of Nonlinear Prefiltering Methods for Efficient and Accurate Surface Shading. IEEE Transactions on Visualization and Computer Graphics 2012;18(2):242–260.

[9] Reeves, WT, Salesin, DH, Cook, RL. Rendering Antialiased Shadows with Depth Maps. In: Proceedings of the ACM SIGGRAPH. 1987, p. 283–291.

[10] Macedo, M, Apolinario, A. Revectorization-Based Shadow Mapping. In: Proceedings of the GI. 2016, p. 75–83.

[11] Donnelly, W, Lauritzen, A. Variance Shadow Maps. In: Proceedings of the ACM I3D. 2006, p. 161–165.

[12] Annen, T, Mertens, T, Seidel, HP, Flerackers, E, Kautz, J. Exponential Shadow Maps. In: Proceedings of GI. 2008, p. 155–161.

[13] Salvi, M. Rendering filtered shadows with exponential shadow maps. In: ShaderX 6.0 Advanced Rendering Techniques. Hingham (Mass.): Charles River Media; 2008, p. 257–274.

[14] Peters, C, Klein, R. Moment Shadow Mapping. In: Proceedings of the ACM I3D. 2015, p. 7–14.

[15] Macedo, M, Apolinario, A. Euclidean Distance Transform Shadow Mapping. In: Proceedings of the GI. 2017, p. 181–189.

[16] Cao, TT, Tang, K, Mohamed, A, Tan, TS. Parallel Banding Algorithm to Compute Exact Distance Transform with the GPU. In: Proceedings of the ACM I3D. 2010, p. 83–90.

[17] Eisemann, E, Schwarz, M, Assarsson, U, Wimmer, M. Real-Time Shadows. Natick, MA, USA: A.K. Peters; 2011. ISBN 978-1568814384.

[18] Woo, A, Poulin, P. Shadow Algorithms Data Miner. Natick, MA, USA: CRC Press; 2012. ISBN 978-1439880234.

[19] Stamminger, M, Drettakis, G. Perspective Shadow Maps. ACM Trans Graph 2002;21(3):557–562.

[20] Lloyd, DB, Govindaraju, NK, Quammen, C, Molnar, SE, Manocha, D. Logarithmic Perspective Shadow Maps. ACM Trans Graph 2008;27(4):106:1–106:32.

[21] Fernando, R, Fernandez, S, Bala, K, Greenberg, DP. Adaptive Shadow Maps. In: Proceedings of the ACM SIGGRAPH. 2001, p. 387–390.

[22] Lauritzen, A, Salvi, M, Lefohn, A. Sample Distribution Shadow Maps. In: Proceedings of the ACM I3D. 2011, p. 97–102.

[23] Bondarev, V. Shadow Map Silhouette Revectorization. In: Proceedings of the ACM I3D. 2014, p. 162–162.

[24] Sen, P, Cammarano, M, Hanrahan, P. Shadow Silhouette Maps. ACM Trans Graph 2003;22(3):521–526.

[25] Pan, M, Wang, R, Chen, W, Zhou, K, Bao, H. Fast, Sub-pixel Antialiased Shadow Maps. Computer Graphics Forum 2009;28(7):1927–1934.

[26] Lecocq, P, Marvie, JE, Sourimant, G, Gautron, P. Sub-pixel Shadow Mapping. In: Proceedings of the ACM I3D. 2014, p. 103–110.

[27] Annen, T, Mertens, T, Bekaert, P, Seidel, HP, Kautz, J. Convolution Shadow Maps. In: Proceedings of the EGSR. 2007, p. 51–60.

[28] Lauritzen, A, McCool, M. Layered Variance Shadow Maps. In: Proceedings of the GI. 2008, p. 139–146.

[29] Gumbau, J, Sbert, M, Szirmay-Kalos, L, Chover, M, González, C. Shadow Map Filtering with Gaussian Shadow Maps. In: Proceedings of the ACM VRCAI. 2011, p. 75–82.

[30] Gumbau, J, Sbert, M, Szirmay-Kalos, L, Chover, M, González, C. Smooth shadow boundaries with exponentially warped gaussian filtering. Computers & Graphics 2013;37(3):214 – 224.

[31] Peters, C. Non-linearly quantized moment shadow maps. In: Proceedings of the HPG. ISBN 978-1-4503-5101-0; 2017, p. 15:1–15:11.

[32] DeCoro, C, Cole, F, Finkelstein, A, Rusinkiewicz, S. Stylized Shadows. In: Proceedings of the ACM NPAR. 2007, p. 77–83.

[33] Saito, T, Takahashi, T. Comprehensible Rendering of 3-D Shapes. In: Proceedings of the ACM SIGGRAPH. 1990, p. 197–206.

[34] Wright, MW, Cipolla, R, Giblin, PJ. Skeletonization using an extended Euclidean distance transform. Image and Vision Computing 1995;13(5):367 – 375.

[35] Hesselink, WH, Roerdink, JBTM. Euclidean Skeletons of Digital Image and Volume Data in Linear Time by the Integer Medial Axis Transform. IEEE Transactions on Pattern Analysis and Machine Intelligence 2008;30(12):2204–2217.

[36] MohammadBagher, M, Kautz, J, Holzschuch, N, Soler, C. Screen-space Percentage-Closer Soft Shadows. In: Proceedings of the ACM SIGGRAPH Posters. 2010, p. 133–133.

[37] Shreiner, D, Sellers, G, Kessenich, JM, Licea-Kane, BM. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3. 8th ed.; Addison-Wesley Professional; 2013. ISBN 0321773039, 9780321773036.

[38] Rost, RJ, Licea-Kane, B, Ginsburg, D, Kessenich, JM, Lichtenbelt, B, Malan, H, et al. OpenGL Shading Language. 3rd ed.; Addison-Wesley Professional; 2009. ISBN 0321637631, 9780321637635.

[39] Kirk, DB, Hwu, WmW. Programming Massively Parallel Processors: A Hands-on Approach. 2 ed.; San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2013. ISBN 9780123914187.

[40] Rong, G, Tan, TS. Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform. In: Proceedings of the ACM I3D. 2006, p. 109–116.

[41] Schneider, J, Kraus, M, Westermann, R. GPU-based real-time discrete euclidean distance transforms with precise error bounds. In: Proceedings of the VISAPP. 2009, p. 435–442.

[42] Wang, J, Tan, Y. Efficient euclidean distance transform algorithm of binary images in arbitrary dimensions. Pattern Recognition 2013;46(1):230 – 242.

[43] McGuire, M. Computer graphics archive. 2017. `https://casual-effects.com/data`.

Appendix A. Algorithm's Pseudocodes

Algorithm 1 Euclidean distance transform shadow mapping

1: $P \leftarrow$ penumbra size;
2: $w_{filter} \leftarrow$ mean filter size;
3: **for** each frame **do**
4:     S $\leftarrow$ renderShadowMap;
5:     G $\leftarrow$ renderGBuffer;
6:     R $\leftarrow$ revectorizeShadow(S, G);
7:     EDT $\leftarrow$ performEDTShadowing(R, G, P);
8:     F $\leftarrow$ filterShadow(EDT, G, w_{filter});
9:     renderShadowedScene(F, G);
10: **end for**

Algorithm 2 Revectorization-based shadow mapping

1: **procedure** revectorizeShadow(S, G)
2:     **for** each surface point **p** in camera view, visible in G **do**
3:         $\tilde{\mathbf{p}} \leftarrow$ transformPointToLightSpace($\mathbf{p}$);
4:         $s \leftarrow$ computeShadowTest(S, $\tilde{\mathbf{p}}_z$);
5:         $\mathbf{N} \leftarrow$ evaluateNeighbourhood(S, $\tilde{\mathbf{p}}_z$);
6:         $\mathbf{d} \leftarrow$ computeDiscontinuityDirections($\mathbf{N}$, s);
7:         $\alpha \leftarrow$ estimateRelativePosition(S, $\mathbf{p}$, $\tilde{\mathbf{p}}$, $\mathbf{d}$);
8:         $v \leftarrow$ computeVisibilityFunction(s, α);
9:     **end for**
10:     **return** an image with v computed for every **p** in G;
11: **end procedure**

In EDTSM, the user is allowed to define the value of two variables: the size of the penumbra to be simulated (Line 1 of Algorithm 1) and the size of the mean filter used to suppress skeleton artifacts in the penumbra (Line 2 of Algorithm 1). Then, for every frame, we need to render two essential textures for EDTSM: the shadow map to allow the real-time shadow computation (Line 4 of Algorithm 1) and the G-buffer to speed up the EDT computation and make it edge aware and invariant to the camera viewpoint (Line 5 of Algorithm 1). Next, on the basis of these two textures, we render another image that contains the anti-aliased hard shadows computed with the improved RBSM (Line 6 of Algorithm 1). Afterwards, we compute the normalized EDT over the image with the revectorized hard shadows to simulate a penumbra with size P (Line 7 of Algorithm 1). In this case, we use the G-buffer previously computed to return the world-space position of each fragment in the camera view, data that allow the EDT to be computed in the world space, rather than in the image space solely. We pass the image with the rendered penumbra to another shader that is responsible for the separable screen-space mean filtering over the penumbra (Line 8 of Algorithm 1). In this filter, the G-buffer is used to detect the depth difference between neighbour fragments, making the mean filtering edge aware. ALso, the G-buffer is used to retrieve the world-space position and adequate the user-defined filter size w_{filter} to be viewpoint invariant. Finally, the resulting filtered image is used to output the final shaded scene (Line 9 of Algorithm 1).

As for RBSM, which is implemented in a single pass on the shader (Algorithm 2), the G-buffer is only used to restrict the hard shadow computation for the visible fragments in the camera view (Line 2 of Algorithm 2). Then, for each visible fragment transformed to the light space (Line 3 of Algorithm 2), the shadow test is computed to determine the hard shadow visibility of the fragment (Line 4 of Algorithm 2) and of its neighbours in the shadow map (Line 5 of Algorithm 2. Afterwards, the directions of where the shadow aliasing is located are detected (Line 6 of Algorithm 2). Finally, a traversal performed in the shadow map allows the estimation of the relative position of each fragment in the aliased boundary (Line 7 of Algorithm 2) and of the final anti-aliased hard shadow visibility condition of each fragment (Line 8 of Algorithm 2). As an output of RBSM, the shader returns an image with the hard shadow intensities computed for every visible fragment in the camera view (Line 10 of Algorithm 2.