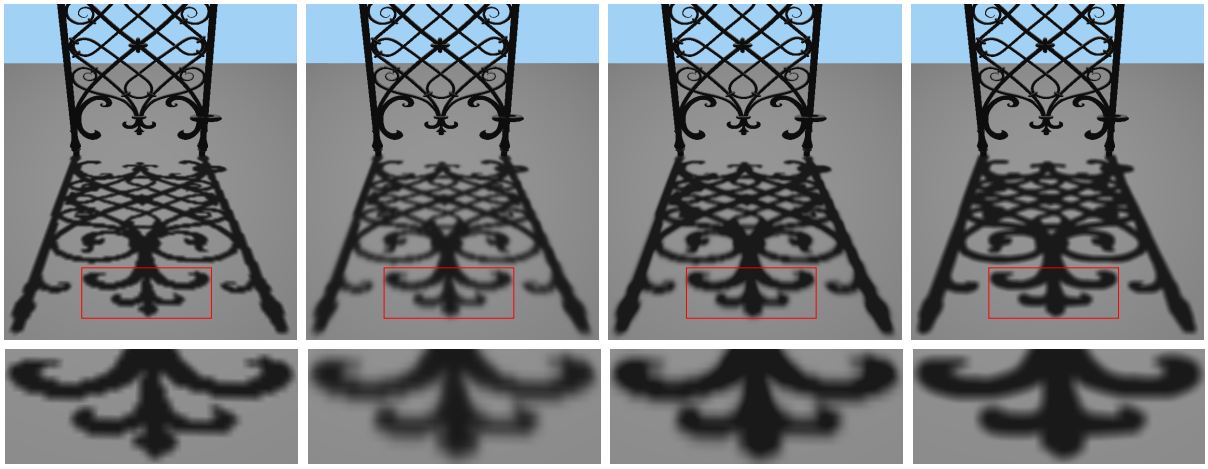
Euclidean Distance Transform Shadow Mapping

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

Federal University of Bahia, Brazil



(a) Percentage-closer filtering (b) Variance shadow mapping (c) Moment shadow mapping (d) EDT shadow mapping

Figure 1: Our Euclidean distance transform (EDT) shadow mapping produces real-time filtered hard shadows with less artifacts than related work. Images were generated for the Fence model using a 1024^2 shadow map resolution.

ABSTRACT

The high-quality simulation of the penumbra effect in real-time shadows is a challenging problem in shadow mapping. The existing shadow map filtering techniques are prone to aliasing and light leaking artifacts which decrease the shadow visual quality. In this paper, we aim to minimize both problems with the Euclidean distance transform shadow mapping. To reduce the perspective aliasing artifacts generated by shadow mapping, we revectorize the hard shadow boundaries using the revectorization-based shadow mapping. Then, an exact normalized Euclidean distance transform is computed in the user-defined penumbra region to simulate the penumbra effect. Finally, a mean filter is applied to further suppress skeleton artifacts generated by the distance transform. The results obtained show that our technique runs entirely on the GPU, produces less artifacts than related work, and provides real-time performance.

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

1 INTRODUCTION

Shadows are of great importance in computer graphics because they improve the realism of the 3D virtual scenes [16, 43]. Unfortunately, real-time, accurate shadow computation still remains an open problem, since the rendering of physically correct shadows is computationally expensive. To simplify the shadow rendering problem, the shadow mapping technique [44] uses a texture called *shadow map* to discretize the 3D space from the light source viewpoint and allow the real-time shadow computation. However, as an image-based approach, the shadows generated by such a technique are prone to aliasing artifacts and temporal incoherence. Moreover, shadow mapping does not simulate the penumbra effect, generating unrealistic hard shadows, rarely seen in the real world.

Texture linear filtering is a common alternative to reduce the aliasing artifacts generated by the texture finite resolution. However, mip-mapping and anisotropic filtering strategies cannot be applied in the shadow map, since the shadow mapping uses a non-linear comparison as shadow test [4]. To solve such a problem, the current shadow map filtering techniques either realize the filtering after the shadow test [26, 32], or change the shadow mapping visibility function to allow shadow map pre-filtering [2, 3, 10, 14, 15, 21, 31, 36]. Such alternatives minimize aliasing artifacts and simulate penumbra in real-time shadows. However, specific problems arise depending on the size of the penumbra simulated. For small penumbra sizes, blurred aliasing artifacts still can be seen in the shadow rendering (see Fig. 1-(a, b, c)). For large penumbra sizes, fine details of the shadow may be suppressed by the shadow overblurring. Also, regardless of the penumbra size, the techniques based on shadow map pre-filtering typically suffer from the light leaking artifacts, where a shadowed region is incorrectly rendered as a fully illuminated region.

In this paper, we present the Euclidean distance transform (EDT) shadow mapping (EDTSM), a new technique which allows efficient shadow filtering and penumbra simulation. Inspired by the recent advances in the field of exact EDT computation on the GPU [6] and efficient shadow anti-aliasing through shadow revectorization [26], we propose a new technique which takes advantage of the revectorized hard shadows to simulate the penumbra effect on the basis of a normalized EDT computation. Then, a mean filter is applied to reduce the typical skeleton artifacts generated by the EDT computation. We show that EDTSM offers real-time performance and does not suffer from light leaking or aliasing artifacts as much as related work.

*e-mail: marciocfmacedo@gmail.com

†e-mail: antonio.apolinario@ufba.br

The remainder of this paper is organized as follows: Sec. 2 briefly describes the most relevant related work in the fields of real-time hard shadow generation and EDT computation. Sec. 3 introduces our EDTSM. In Sec. 4, we discuss the experimental results, comparing our approach with related work in terms of visual quality and rendering performance. Finally, Sec. 5 gives the concluding remarks and key directions for future work.

2 Related Work

In this section, we focus on the review of relevant related work proposed in the fields of shadow anti-aliasing, shadow map filtering, and EDT computation. An in-depth review of the existing shadow rendering algorithms is beyond the scope of this paper. We refer the reader to the books of Eisemann et al. [11] and Woo et al. [46]. For a complete review of the existing techniques for EDT, see [12, 19].

Shadow Anti-Aliasing: Shadow mapping [44] is an image-based shadow algorithm composed of two passes. In the first pass, the technique samples the 3D space viewed from the light source and rasterizes the distance of the light source to the closest surface points of the scene into a depth texture called shadow map. In the second pass, each surface point visible in the camera view is projected into the light source view and its distance is compared to the one stored in the shadow map to determine whether the surface point is lit or in shadow (i.e., shadow test). Due to the limited shadow map resolution, shadow mapping may generate aliasing artifacts along the shadow silhouette.

One way to reduce shadow aliasing artifacts relies on the reparametrization of the shadow map generation by the use of warping strategies. The most common techniques [25, 27, 40, 45] use the post-perspective space to render the shadow map, improving the shadow map resolution by focusing the sampling on the parts of the scene most relevant for the shadow rendering. Indeed, these techniques minimize the shadow aliasing artifacts, provide real-time performance, but suffer from shadow flickering artifacts caused by the camera or light source movement. Also, they improve the shadow map resolution, but are not able to completely reduce the shadow aliasing artifacts.

Another approach to minimize shadow aliasing consists in the partitioning of the view frustum, such that a shadow map is associated for each partitioned space [49, 50]. Sample distribution shadow mapping [22], for instance, splits the view frustum along the z-axis and generates tight shadow maps for each partitioned space. Rather than partitioning the 3D space, other techniques evaluate the aliasing error caused by the use of a single low-resolution shadow map, and change the shadow map configuration adaptively (either by increasing the shadow map resolution, generating more shadow maps) in order to minimize the shadow aliasing [13, 17, 24]. The partitioning strategy is useful to generate high-quality shadows for large-scale scenarios. On the other hand, the additional cost caused by the partitioning may be inadequate for small-scale scenarios.

To reduce the aliasing artifacts, several techniques store additional information of the light blocker geometry into the shadow map [30, 38] to improve the shadow test, or make use of irregular data structures to provide a more accurate sampling of the shadow map [1, 18]. In general, these techniques effectively reduce the shadow aliasing artifacts, at the cost of more memory consumption and processing time, although more recent, efficient solutions do exist [23, 26, 48].

Shadow Map Filtering: Most of the techniques mentioned so far are able to provide shadow anti-aliasing, but they cannot reproduce the penumbra effect of the shadow, since they use a binary visibility function to compute the hard shadows.

Percentage-closer filtering (PCF) [32] minimizes aliasing and simulates penumbra by taking the average of shadow test results over a sampled region. PCF is free from light leaking artifacts, but is not scalable with respect to the filter size, requires several samples to reduce banding artifacts and does not support pre-filtering.

Variance shadow mapping (VSM) [10] is a statistical approach which stores depth and squared depth into the shadow map and use such an information to determine the probability of whether a point is in shadow or is lit. VSM supports pre-filtering and is scalable, at the expense of the light leaking artifacts generation.

Convolution shadow mapping (CSM) [2] linearizes the shadow test by approximating it as an expansion of Fourier series. The shadow map is converted into several basis textures, which are filtered and used to perform shadow anti-aliasing. The final shadow intensity is computed by a weighted sum of the basis functions stored in the basis textures. CSM supports pre-filtering and does not suffer from light leaking as much as VSM. However, this technique demands higher memory footprint and processing time than the alternative shadow map pre-filtering techniques.

Exponential shadow mapping (ESM) [3, 36] approximates the shadow test by an exponential function. The depth values stored in the shadow map are converted into exponent-transformed depth values, which are used to determine the final shadow intensity. ESM is an alternative to VSM, but still prone to light leaking.

Exponential variance shadow mapping (EVSM) [21] merges ESM and VSM theories to produce high-quality anti-aliasing and penumbra simulation. In this case, light leaking only occurs at the fragments where such an artifact was present in both ESM and VSM.

Gaussian shadow mapping [14, 15] is another statistical filtering technique which uses the Gaussian cumulative distribution function to estimate the shadow intensity. Similarly to EVSM, an alternative, hybrid visibility function is used with the exponential function to further reduce light leaking artifacts.

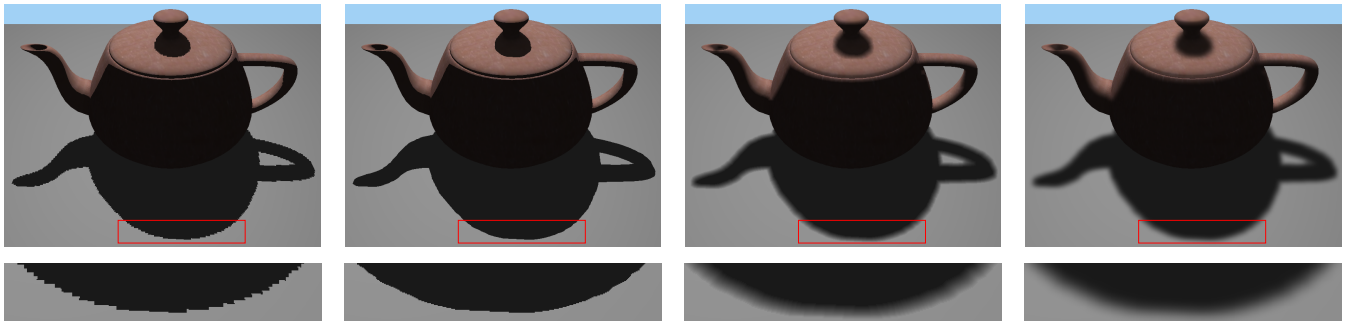
Moment shadow mapping (MSM) [31] stores depth raised from the first to the fourth power in the shadow map and estimates the shadow intensity according to the Hamburger or Hausdorff moment problem. As an improved version of the VSM technique, MSM greatly reduces the light leaking artifacts obtained in the VSM pre-filtering approach.

Revectorization-based PCF (RPCF) [26] first generates anti-aliased hard shadows by the revectorization of the jagged shadow silhouette, then applies the PCF over the revectorized hard shadows to fake the penumbra. Indeed, RPCF improves the accuracy of PCF, requires just a few samples to reduce banding artifacts, and works well for low-resolution shadow maps, generating fake penumbra free from aliasing artifacts. Unfortunately, RPCF is slower than PCF for the same kernel sizes.

Shadow map filtering techniques are an efficient alternative to reduce aliasing artifacts and fake penumbra for the hard shadows produced by shadow mapping. Pre-filtering techniques are scalable in terms of filter size, but suffer from light leaking. On the other hand, PCF and RPCF are not prone to light leaking, but are not as well scalable. Most of these techniques produce blurred jagged shadows for small penumbra sizes when used with shadow maps of low resolution. For large penumbra sizes, they may suffer from banding artifacts if an insufficient kernel size is used. Also, details of the shadow silhouette may be lost due to the shadow blurring. In this paper, we want to solve these problems by computing the penumbra intensities as a normalized EDT over the penumbra region of the shadow, as we will detail in Sec. 3.

Euclidean Distance Transform: EDT is used in several applications, such as image processing, computer vision, computer graphics, and scientific visualization. In our context, the distance transform is defined as an operation that receives as input a binary image and returns as output a gray-level image, in which the intensity of each pixel is computed as the distance from the pixel to the closest foreground pixel (termed *site*) in the binary image.

Vector propagation [8] is one of the most common techniques used to compute fast, approximate EDTs. In this method, vector



(a) Aliased Shadow (b) Revectorized Shadow (c) Normalized EDT (d) Filtered EDT

Figure 2: An overview of the EDT shadow mapping. Given the aliased hard shadows obtained with the traditional shadow mapping (a), revectorization-based shadow mapping is applied to generate anti-aliased hard shadows (b). Then, penumbra intensities are computed on the basis of a normalized EDT over the penumbra region (c). Finally, a mean filter (d) is applied over the normalized EDT to reduce skeleton artifacts generated by the EDT computation. Images were generated for the Teapot model using a 1024^2 shadow map resolution.

templates are propagated and swept through the image to compute accurate EDT. However, the original technique is inadequate for our purposes since it is CPU-based and does not run in real time. In fact, even the fastest CPU-based solution [42] proposed to compute EDT achieves only interactive performance for high-resolution images.

Jump flooding algorithm (JFA) [33] adapts the vector propagation method to the GPU by performing it in several rounds, with different step sizes, each pixel realizing its own processing in parallel. This technique is faster than the original vector propagation method, but produces approximate EDT, susceptible to errors. The fast hierarchical algorithm proposed by Cuntz and Kolb [7] downsamples and upsamples the original image to speed up the JFA, at the cost of a low-accurate EDT generation.

The approach proposed by Schneider et al. [37] modifies the vector templates proposed in [8], such that the vector propagation can be efficiently implemented on the GPU. This approach is, in general, slower than JFA, but produces EDT close to the exact solution.

Different from the vector propagation, these techniques run on the GPU, but provide interactive performance and do not produce accurate EDT, being unsuitable to be used in a real-time hard shadow technique.

Parallel banding algorithm (PBA) [6] is the first technique to compute exact EDT in GPU in real time. The main idea of PBA is to divide the EDT processing into bands, which are processed in parallel by the GPU. In the first pass of the technique, each row of the image is divided into bands, in which a two-pass horizontal sweeping is performed, resulting in the 1D EDT computation. Next, each column of the image is divided into bands, in which a vertical sweeping is performed. Finally, each pixel of the image can localize its closest site and compute the EDT. PBA is faster and more accurate than related work.

In this sense, due to the recent advances in the field of EDT, like the proposition of the PBA, we propose a new technique which simulates the penumbra effect using a normalized EDT.

To the best of our knowledge, the stylized shadows method [9] is the only existing technique which uses a distance transform for shadow computation. In this technique, a signed distance function is computed from an accurate hard shadow and is used to guide the shadow blur, which allows the generation of non-photorealistic shadows. Four parameters are used to control the shadow rendering: inflation, to control the shadow size; brightness, to control the shadow intensity; softness, to control the penumbra size; and abstraction, to control the shadow accuracy. The distance transform is used for both inflation and softness parameters, helping in the control of both shadow and penumbra sizes.

Similar to our approach, the stylized shadows method uses the world-space distance metric to compute the EDT, making the approach viewpoint invariant. Different from our approach, such a technique uses a blur filter to modify the shadow's shape, does not run in real time for dynamic scenes, and was developed for non-photorealistic shadow rendering, to help artists with the task of shadow modelling. Although we also do not compute physically correct shadows, we intend to solve the problems currently found in the field of filtered hard shadow generation, providing real-time performance, competitive against related work. Also, we use a blur filter aiming to suppress skeleton artifacts generated by the distance transform computation.

In this paper, we show that our method, which operates on the screen space, similarly to other existing techniques (e.g., [5, 29]) reduces perspective, light leaking and banding artifacts even for small penumbra sizes, achieving high-quality anti-aliasing with real-time performance.

3 Euclidean Distance Transform Shadow Mapping

In this section, we introduce the EDTSM. An overview can be seen in Fig. 2 and is described in more details in the next subsections.

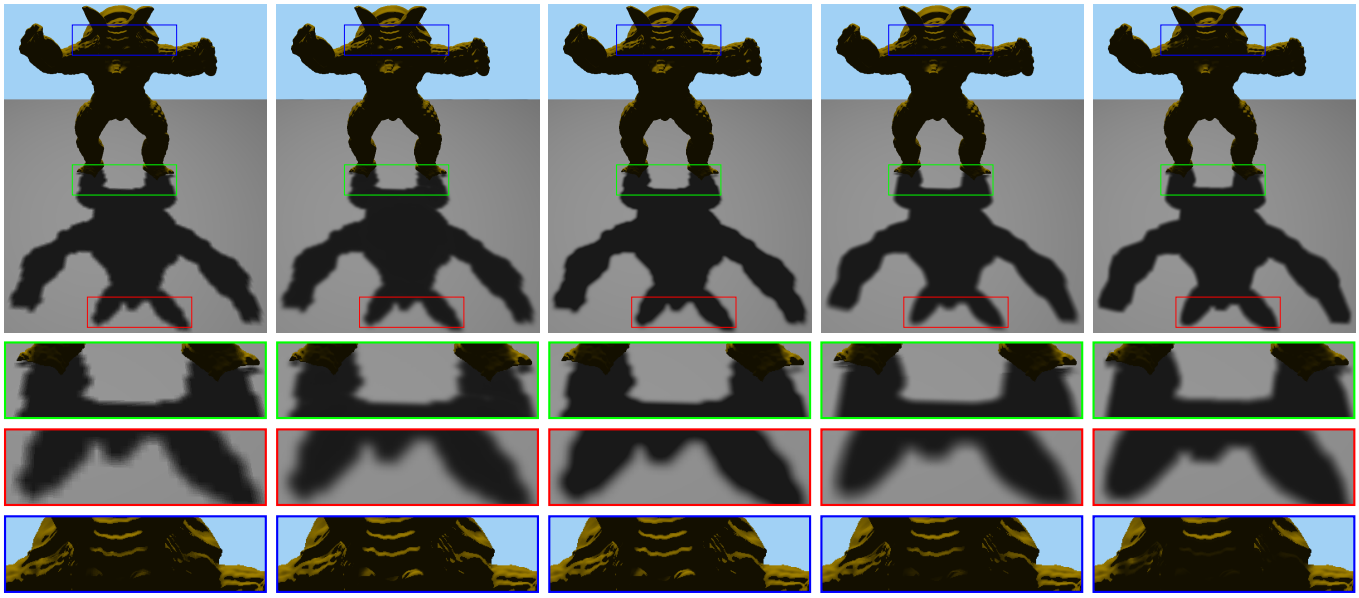
3.1 Shadow Map Rendering

The first step of the EDTSM consists in the generation of the shadow map texture, as seen from the light source view. The shadow map texture allows us to compute hard shadows, as proposed in the shadow mapping technique [44].

3.2 Shadow Revectorization

Shadow mapping produces shadows with aliasing artifacts along the shadow silhouette (Fig. 2-(a)), mainly for shadow maps with low resolution. To reduce such an artifact and produce high-quality hard shadows at little additional cost, we use the single-pass shadow map silhouette revectorization [26] to compute the hard shadows using the shadow map computed in the previous step (Fig. 2-(b)).

In this technique, jagged shadow silhouettes are detected in the light space according to the difference between shadow test results of neighbour shadow map texels. Then, for each fragment inside the shadow silhouette, a traversal is performed in the light space to determine the size of the shadow silhouette and the relative position of the fragment inside it. In the camera space, such a relative position is used in a linear comparison which determines the revectorized shadow silhouette.



(a) PCF (b) VSM (c) MSM (d) RPCF (e) EDTSM

Figure 3: Filtered hard shadows produced by different techniques. Green closeups show how the techniques handle aliasing artifacts and light leaking on contact regions. Red closeups show an additional view with respect to the shadow aliasing artifacts. Blue closeups show whether the techniques suffer from shadow overestimation. Images were generated for the Armadillo model using a 512^2 shadow map resolution.

To restrict shadow and shading computation to the fragments visible to the camera, as suggested by the deferred rendering pipeline, we render the G-Buffer [35] of the scene viewed from the camera and use it in the next steps of the algorithm.

3.3 EDT Shadowing

Once we have generated the revectorized hard shadows, we can compute the penumbra intensities according to a normalized EDT (Fig. 2-(c)). In this step, our main goal is to compute, for each penumbra fragment, its Euclidean distance to the closest shadow silhouette fragment in the camera view, to normalize such a distance to the closed interval $[0,1]$, where 0 is the value associated with an umbra fragment, and 1 is the value associated with a lit fragment.

Let us define as sites the fragments located at the shadow silhouette. To locate the sites in the screen space, we apply a 3×3 non-separable rectangular filter kernel over the revectorized shadows and label a fragment as a site if there is a difference between the visibility condition of the current fragment and one of its neighbours in the filter kernel. Then, for each non-site fragment, we compute the EDT, which returns the Euclidean distance (D) of the fragment to the closest site.

Let us denote P a user-defined parameter which determines the desired penumbra size. Half of the penumbra size ($\frac{P}{2}$) belongs to the interior (shadowed) side of the shadow edge, and the other half belongs to the exterior (lit) side of the shadow edge. Then, for every fragment inside the penumbra region (i.e., $D \leq \frac{P}{2}$), we can compute its penumbra intensity (I) as

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

From Equation 1, shadowed and lit fragments whose distance to the shadow silhouette is $\frac{P}{2}$ have the new intensities 0 and 1, respectively. Each fragment whose distance to the shadow silhouette is lower than $\frac{P}{2}$ has a new penumbra intensity which lies between 0 and 1.

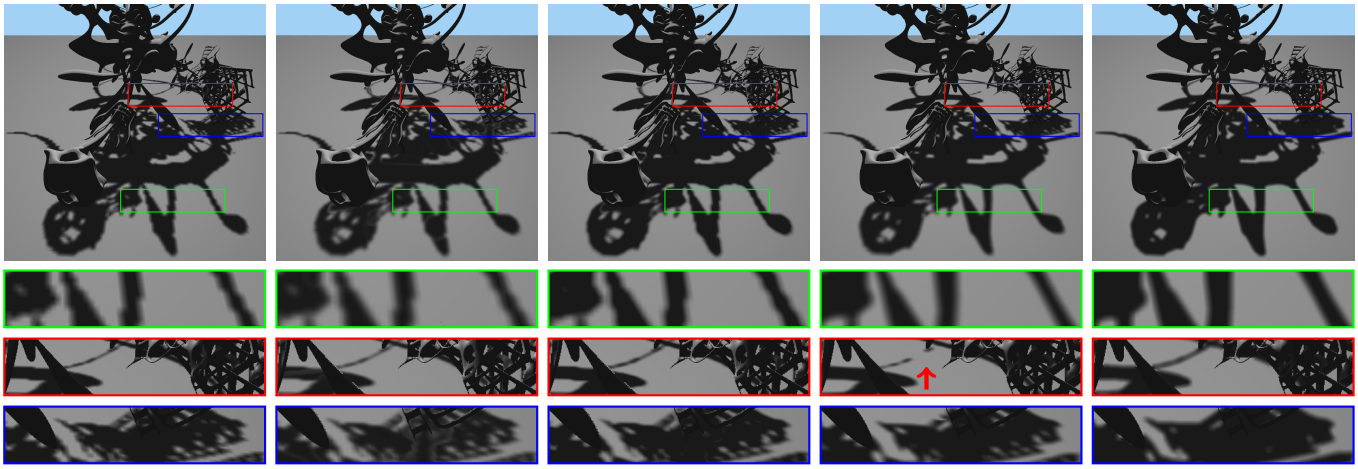
Since we perform this step in the screen space, we lose information about edge location (i.e., boundaries between objects) and we may generate different penumbra sizes according to the distance of the viewer to the scene. To solve the first problem, we take into consideration the depth buffer of the scene as seen from the camera, which was previously stored in the G-Buffer, to detect the edges of the objects. In this sense, a fragment is only considered to be in penumbra if the depth difference between the fragment and its closest site is below a user-defined threshold. To produce a viewpoint-invariant screen-space penumbra size estimation, we compute the Euclidean distance values with respect to the world space, rather than using only the screen-space information.

3.4 EDT Filtering

After the EDT shadow computation, skeleton artifacts (Fig. 2-(c)) arise along the singularities (i.e., gradient discontinuities) in the distance transform [47]. To minimize such artifacts, we apply a screen-space separable rectangular mean filter kernel over the shadows previously computed (Fig. 2-(d)). We have chosen the mean filtering for this step because it is simple to implement, fast, separable and effectively minimizes the skeleton artifacts even for low-order kernel sizes. Gaussian filtering could be used as an alternative in this case, but we have found that the mean filtering is sufficient for our purposes. Furthermore, since we do not need to keep sharp boundaries in the filtered hard shadow, we have opted to not use the bilateral filtering [41].

Similarly to the EDT shadowing algorithm (Sect. 3.3), the EDT filtering needs to be edge aware and viewpoint invariant. We solve the first problem using the same solution shown in Sect. 3.3, taking into account depth difference when applying the blur filter. Inspired by the screen-space soft shadow algorithms, we solve the second problem by estimating the variable mean filter size w_{filter}^{screen} for each fragment [29]

$$w_{filter}^{screen} = \frac{w_{filter} z_{screen}}{z_{eye}}, \quad (2)$$



(a) PCF (b) VSM (c) MSM (d) RPCF (e) EDTSM

Figure 4: Filtered hard shadows produced by different techniques. Green closeups show whether the techniques handle aliasing artifacts. Red closeups show how the techniques perform under fine details. Blue closeups show whether the techniques suffer from light leaking artifacts. Images were generated for the YeahRight model using a 1024^2 shadow map resolution.

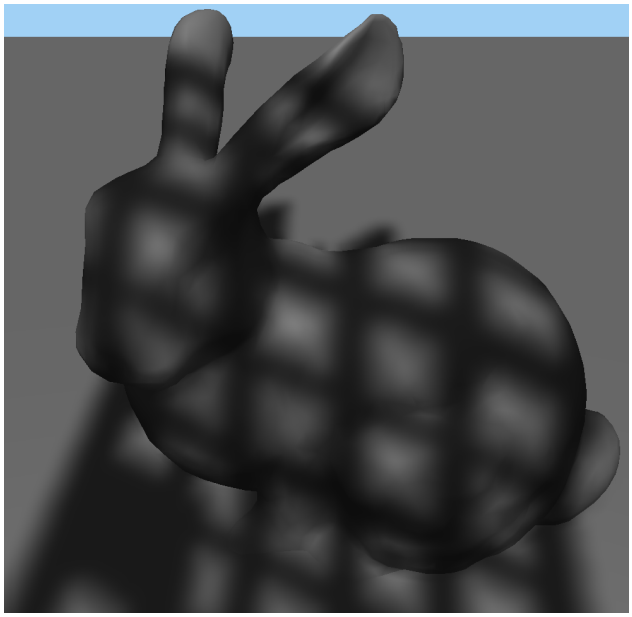


Figure 5: EDTSM supports shadow rendering on a non-planar receiver. Image was generated for the Door (light blocker) and Bunny (shadow receiver) models using a 1024^2 shadow map resolution.

$$z_{screen} = \frac{1}{2\tan\frac{fov}{2}}, \tag{3}$$

where fov specifies the field of view angle, z_{eye} is the distance of the fragment to the camera and w_{filter} is the user-defined mean filter size. By computing w_{filter}^{screen}, which varies according to the distance of the camera to the scene, we improve the temporal coherency of the EDT filtering.

4 Results and Discussion

In this section, we evaluate the hard shadow filtering techniques in terms of visual quality and performance. All the tests were executed on an Intel Core™ i7-3770K CPU (3.50 GHz), 8GB RAM, and an NVIDIA GeForce GTX Titan X graphics card. EDTSM was implemented using OpenGL [39] and GLSL [34] languages. To compute the EDT, we have used the original implementation of the PBA [6], because, to the best of our knowledge, it is the only one which computes exact EDT in real time on the GPU. Since the PBA is implemented in CUDA [20], we used the CUDA/OpenGL interoperability to optimize resource management.

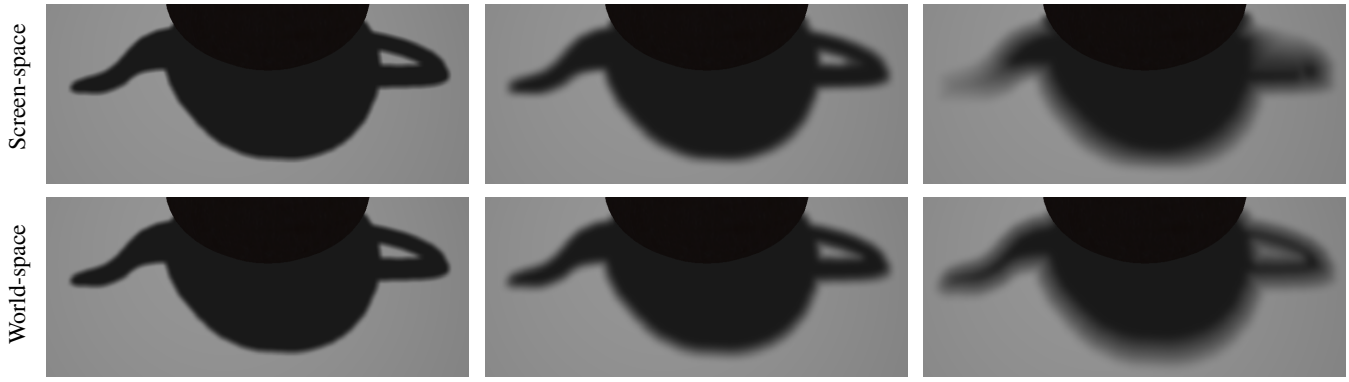
We have used a kernel of size 15×15 to suppress skeleton and banding artifacts for the filtering techniques. Specifically for RPCF, we have used a kernel of size 7×7 since this algorithm requires less samples to reduce the banding artifacts [26]. In the accompanying video, we show additional results of the EDTSM, including temporal coherency.

4.1 Visual Quality

We compared the filtered hard shadows computed using our approach with two of the most traditional shadow map filtering techniques (PCF and VSM) and two of the most recent ones (MSM and RPCF), as shown in Fig. 3 and Fig. 4.

The non-revectorization-based filtering techniques (PCF, VSM and MSM) have aliasing artifacts along the fake penumbra for shadow maps of low resolution (as shown in Fig. 3 and the green closeups in Fig. 4-(a, b, c)). RPCF is not able to handle the fine details if an insufficient shadow map resolution is used (as pointed by the red arrow in Fig. 4-(d)). Pre-filtering techniques, such as VSM, are prone to several light leaking artifacts, as shown in the green closeup of Fig. 3-(b) and the blue closeup of Fig. 4-(b). In this case, MSM reduces significantly the light leaking artifacts, as seen in the blue closeup of Fig. 4-(c). Our EDTSM does not suffer from aliasing artifacts (green and red closeups in Fig. 3-(e) and the green closeup in Fig. 4-(e)), shadow holes as much as RPCF (red closeup in Fig. 4-(e)) and light leaking artifacts (green closeup in Fig. 3-(e), and blue closeup in Fig. 4-(e)). However, details of the shadow silhouette may be lost due to the artifact of shadow overestimation.

In EDTSM, shadow overestimation may be caused by two factors. The first is the mean filtering, whose blurring enlarges the entire



(a) Small penumbra size (b) Medium penumbra size (c) Large penumbra size

Figure 6: A comparison of different implementations of the EDTSM for different penumbra sizes. Images were generated for the Teapot model using a 1024^2 shadow map resolution.

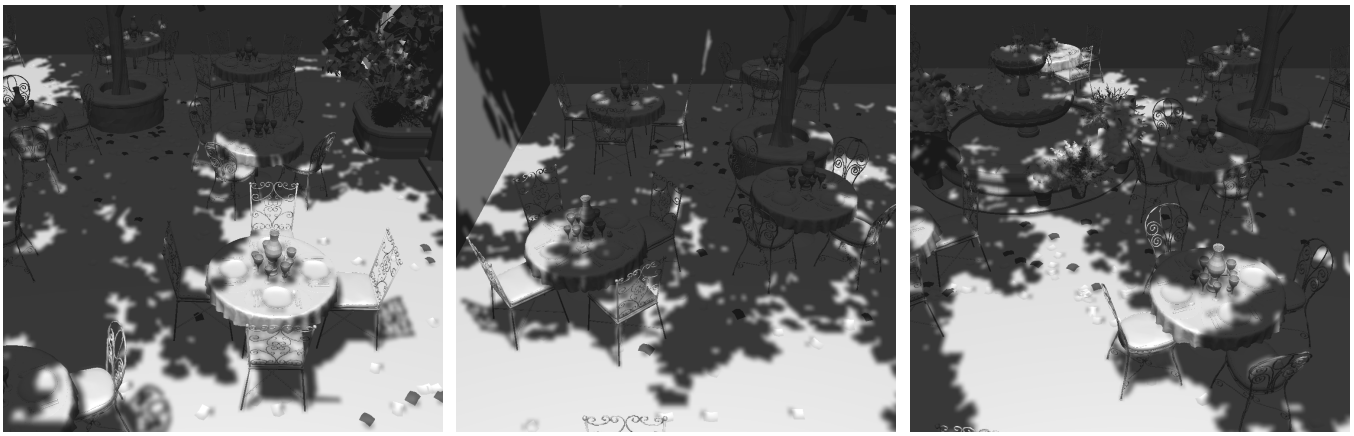


Figure 7: EDTSM works well for more complex scenarios with multiple objects. The image contains different viewpoints of the San Miguel model, rendered using a 1024^2 shadow map resolution.

shadow region. This is a special problem for lit regions which are surrounded by shadow, since they will be incorrectly put in the fake penumbra. Also, as shown in the blue closeup of Fig. 3-(e), if distinct parts of the same object do not have much depth difference between them, our edge-aware filtering approach assumes that they belong the same region and performs the blurring erroneously. The second factor which may cause shadow overestimation is the hard shadow revectorization, which puts small closed lit regions entirely in shadow, as shown in the blue closeup in Fig. 4-(e). While such an artifact may generate fake penumbra different from related work, we show in the accompanying video that our approach is temporally coherent. Therefore, the shadow is overestimated, but does not cause flickering artifacts during animation.

Without any modification of the algorithm shown in Sec. 3, EDTSM is able to fake penumbra for shadows cast on planar receivers (as shown in most of the figures contained in this paper), as well as non-planar receivers (Fig. 5).

In Fig. 6, we present a comparison between two distinct implementations of the EDTSM approach: one which uses the screen-space distance metric to estimate the penumbra size, compute and filter the EDT (Fig. 6-top), and the one proposed in this paper, which uses the world-space distance metric to compute the fake penumbra (Fig. 6-bottom). While both of them perform well for a small penumbra size (Fig. 6-(a)), artifacts may arise for medium and large penumbra sizes when using the screen-space distance metric (Fig. 6-(b, c)). In this sense, the use of the world-space metric allows not only the viewpoint-invariant rendering of the fake penumbra, but also minimizes the incidence of potential artifacts located at the shadow silhouette.

As shown in Fig. 7, EDTSM supports the shadow rendering for game-like scenarios, with several light blocker and shadow receiver objects. Unfortunately, as can be seen along the thin legs of some chairs in Fig. 7, light leaking artifacts may arise in the final rendering due to the small depth difference between different objects, which affects the EDT computation.

4.2 Rendering Time

In Table 1, Table 2 and Table 3, we compare the performance obtained by our approach and related work. Only the scenarios shown in Fig. 1, Fig. 3 and Fig. 4 were evaluated, because they are the only ones where we compare our approach with related work in terms of visual quality. Also, we have presented the performance of VSM and MSM into a single row in the tables, because we have obtained similar performance for them.

RPCF provides one of the best visual quality results, but is the slowest technique, independent of the scene configuration used. EDTSM is slightly slower than PCF, VSM and MSM, mainly when comparing both techniques for the same kernel size and high output resolutions. Nevertheless, it is worthy to note that, by using the concept of separable filtering, EDTSM is faster than PCF for high

Scene	Method	Shadow Map Resolution			
		512^2	1024^2	2048^2	4096^2
Fig. 1	PF	4.1 ms	4.3 ms	4.5 ms	5.5 ms
	PCF	5.0 ms	5.1 ms	5.2 ms	5.6 ms
	EDTSM	6.3 ms	6.4 ms	6.5 ms	7.4 ms
	RPCF	22.2 ms	22.7 ms	23.2 ms	27.7 ms
Fig. 3	PF	4.7 ms	4.8 ms	5.1 ms	6.3 ms
	PCF	5.3 ms	5.4 ms	5.6 ms	6.4 ms
	EDTSM	6.4 ms	6.5 ms	6.7 ms	7.5 ms
	RPCF	12.9 ms	13.6 ms	15.2 ms	17.8 ms
Fig. 4	PF	10.5 ms	10.6 ms	10.8 ms	12.0 ms
	PCF	11.3 ms	11.4 ms	11.4 ms	11.7 ms
	EDTSM	12.8 ms	12.9 ms	13.0 ms	13.7 ms
	RPCF	26.2 ms	27.3 ms	27.8 ms	30.0 ms

Table 1: Processing time for several hard shadow filtering techniques and different scenes rendered at an output HD resolution. Measurements include varying shadow map resolution. PF - Pre-filtering techniques (namely VSM and MSM).

Scene	Method	Output Resolution		
		SD	HD	Full-HD
Fig. 1	PF	3.2 ms	4.3 ms	4.8 ms
	PCF	3.2 ms	5.1 ms	5.3 ms
	EDTSM	4.1 ms	6.4 ms	8.0 ms
	RPCF	10.6 ms	22.7 ms	25.0 ms
Fig. 3	PF	3.2 ms	4.7 ms	5.7 ms
	PCF	3.2 ms	5.4 ms	5.8 ms
	EDTSM	4.3 ms	6.4 ms	8.1 ms
	RPCF	7.1 ms	12.9 ms	16.6 ms
Fig. 4	PF	9.9 ms	10.6 ms	11.3 ms
	PCF	9.8 ms	11.4 ms	11.9 ms
	EDTSM	10.7 ms	12.9 ms	14.7 ms
	RPCF	16.4 ms	27.3 ms	30.3 ms

Table 2: Processing time for several hard shadow filtering techniques. Fig. 1, Fig. 3, and Fig. 4 were rendered using 1024^2, 512^2 and 1024^2 shadow map resolutions, respectively. Measurements include varying viewport resolution. PF - Pre-filtering techniques (namely VSM and MSM).

Scene	Method	Kernel Size			
		7×7	15×15	23×23	31×31
Fig. 1	PF	3.9 ms	4.3 ms	4.5 ms	4.7 ms
	PCF	3.4 ms	5.1 ms	7.4 ms	10.2 ms
	EDTSM	5.9 ms	6.4 ms	6.8 ms	7.2 ms
	RPCF	22.2 ms	76.9 ms	142.8 ms	200.0 ms
Fig. 3	PF	4.5 ms	4.7 ms	5.1 ms	5.3 ms
	PCF	3.5 ms	5.4 ms	7.5 ms	10.5 ms
	EDTSM	6.2 ms	6.4 ms	6.7 ms	7.0 ms
	RPCF	12.9 ms	39.6 ms	89.2 ms	142.8 ms
Fig. 4	PF	9.8 ms	10.6 ms	10.7 ms	11.1 ms
	PCF	9.8 ms	11.4 ms	13.5 ms	17.0 ms
	EDTSM	12.3 ms	12.9 ms	13.5 ms	14.2 ms
	RPCF	26.2 ms	77.5 ms	166.6 ms	285.7 ms

Table 3: Processing time for several hard shadow filtering techniques rendered at an output HD resolution. Fig. 1, Fig. 3, and Fig. 4 were rendered using 1024^2, 512^2 and 1024^2 shadow map resolutions, respectively. Measurements include varying kernel size. PF - Pre-filtering techniques (namely VSM and MSM).

order filter sizes (Table 3).

5 Conclusions and Future Work

In this paper, we have proposed the Euclidean distance transform shadow mapping, a technique which uses the notions of distance transform and shadow revectorization to produce fake penumbra for real-time shadows. From an analysis of the results, EDTSM outperforms related work in terms of visual quality, mainly for low-resolution shadow maps. Also, EDTSM is faster than RPCF and more scalable than PCF for high order filter sizes. Hence, EDTSM becomes an alternative to be used in games and other applications where the visual quality or processing time could be improved with our algorithm.

Unfortunately, EDTSM is prone to shadow overestimation caused by the EDT filtering and the shadow revectorization. The use of an alternative filtering algorithm which effectively minimizes the skeleton artifacts without causing overblurring would be useful in this case. Likewise, the replacement of the shadow revectorization by another technique which computes real-time anti-aliased hard shadows without shadow overestimation would minimize such an artifact.

In terms of performance, EDTSM is slightly slower than related work for the same scene configuration. Since the EDT computation is the step which consumes most of the processing time in our approach, the proposition of a more efficient GPU-based EDT algorithm which computes exact, or even approximate EDT, would speed up our approach. For future work, we also intend to extend the concept of EDTSM for visually plausible or even accurate soft shadow computation.

Acknowledgments

We wish to thank the authors of [6] for sharing the code of the parallel banding algorithm. Also, we would like to thank the NVIDIA Corporation for providing the graphics hardware used in the tests, through the GPU Education Center program in 2016. YeahRight model is a courtesy of Keenan Crane. San Miguel model is a courtesy of Morgan McGuire and Guillermo Llaguno [28]. This research is financially supported by Coordenação de Aperfeiçoamento de Pessoal do Nível Superior (CAPES).

References

[1] T. Aila and S. Laine. Alias-Free Shadow Maps. In *Proceedings of the EGSR*. The Eurographics Association, Aire-la-Ville, Switzerland, 2004.

[2] T. Annen, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz. Convolution Shadow Maps. In J. Kautz and S. Pattanaik, eds., *Proceedings of the EGSR*, pp. 51–60. The Eurographics Association, Aire-la-Ville, Switzerland, 2007.

[3] T. Annen, T. Mertens, H.-P. Seidel, E. Flerackers, and J. Kautz. Exponential Shadow Maps. In *Proceedings of GI*, pp. 155–161. Canadian Information Processing Society, Toronto, Ont., Canada, 2008.

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

[5] J. M. Buades, J. Gumbau, and M. Chover. Separable soft shadow mapping. *The Visual Computer*, 32(2):167–178, 2015.

[6] T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan. Parallel Banding Algorithm to Compute Exact Distance Transform with the GPU. In *Proceedings of the ACM I3D*, pp. 83–90. ACM, New York, NY, USA, 2010.

[7] N. Cuntz and A. Kolb. Fast Hierarchical 3D Distance Transforms on the GPU. In *Proceedings of the ACM EUROGRAPHICS*. The Eurographics Association, 2007.

[8] P.-E. Danielsson. Euclidean Distance Mapping. *Computer Graphics And image Processing*, 14:227–248, 1980.

[9] C. DeCoro, F. Cole, A. Finkelstein, and S. Rusinkiewicz. Stylized Shadows. In *Proceedings of the ACM NPAR*, pp. 77–83. ACM, New York, NY, USA, 2007.

[10] W. Donnelly and A. Lauritzen. Variance Shadow Maps. In *Proceedings of the ACM I3D*, pp. 161–165. ACM, New York, NY, USA, 2006.

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

[12] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno. 2D Euclidean Distance Transform Algorithms: A Comparative Survey. *ACM Comput. Surv.*, 40(1):2:1–2:44, 2008.

[13] R. Fernando, S. Fernandez, K. Bala, and D. P. Greenberg. Adaptive Shadow Maps. In *Proceedings of the ACM SIGGRAPH*, pp. 387–390. ACM, New York, NY, USA, 2001.

[14] J. Gumbau, M. Sbert, L. Szirmay-Kalos, M. Chover, and C. González. Shadow Map Filtering with Gaussian Shadow Maps. In *Proceedings of the ACM VRCAI*, pp. 75–82. ACM, New York, NY, USA, 2011.

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

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

[17] N. Jia, D. Luo, and Y. Zhang. Distorted Shadow Mapping. In *Proceedings of the ACM VRST*, pp. 209–214. ACM, New York, NY, USA, 2013.

[18] G. S. Johnson, J. Lee, C. A. Burns, and W. R. Mark. The irregular Z-buffer: Hardware acceleration for irregular data structures. *ACM Trans. Graph.*, 24(4):1462–1482, Oct. 2005.

[19] M. W. Jones, J. A. Baerentzen, and M. Sramek. 3D Distance Fields: A Survey of Techniques and Applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599, 2006.

[20] D. B. Kirk and W.-m. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 ed., 2013.

[21] A. Lauritzen and M. McCool. Layered Variance Shadow Maps. In *Proceedings of the GI*, pp. 139–146. Canadian Information Processing Society, Toronto, Ont., Canada, 2008.

[22] A. Lauritzen, M. Salvi, and A. Lefohn. Sample Distribution Shadow Maps. In *Proceedings of the ACM I3D*, pp. 97–102. ACM, New York, NY, USA, 2011.

[23] P. Lecocq, J.-E. Marvie, G. Sourimant, and P. Gautron. Sub-pixel Shadow Mapping. In *Proceedings of the ACM I3D*, pp. 103–110. ACM, New York, NY, USA, 2014.

[24] A. E. Lefohn, S. Sengupta, and J. D. Owens. Resolution-matched Shadow Maps. *ACM Trans. Graph.*, 26(4), Oct. 2007.

[25] D. B. Lloyd, N. K. Govindaraju, C. Quammen, S. E. Molnar, and D. Manocha. Logarithmic Perspective Shadow Maps. *ACM Trans. Graph.*, 27(4):106:1–106:32, Nov. 2008.

[26] M. Macedo and A. Apolinario. Revectorization-Based Shadow Mapping. In *Proceedings of the GI*, pp. 75–83. Canadian Information Processing Society, Toronto, Ont., Canada, 2016.

[27] T. Martin and T.-S. Tan. Anti-aliasing and Continuity with Trapezoidal Shadow Maps. In *Proceedings of the EGSR*, pp. 153–160. Eurographics Association, Aire-la-Ville, Switzerland, 2004.

[28] M. McGuire. Computer graphics archive, August 2011. `http://graphics.cs.williams.edu/data`.

[29] M. MohammadBagher, J. Kautz, N. Holzschuch, and C. Soler. Screen-space Percentage-Closer Soft Shadows. In *Proceedings of the ACM SIGGRAPH Posters*, pp. 133–133. ACM, New York, NY, USA, 2010.

[30] 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.

[31] C. Peters and R. Klein. Moment Shadow Mapping. In *Proceedings of the ACM I3D*, pp. 7–14. ACM, New York, NY, USA, 2015.

[32] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering Antialiased Shadows with Depth Maps. In *Proceedings of the ACM SIGGRAPH*, pp. 283–291. ACM, New York, NY, USA, 1987.

[33] G. Rong and T.-S. Tan. Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform. In *Proceedings of the ACM I3D*, pp. 109–116. ACM, New York, NY, USA, 2006.

[34] R. J. Rost, B. Licea-Kane, D. Ginsburg, J. M. Kessenich, B. Lichtenbelt, H. Malan, and M. Weiblen. *OpenGL Shading Language*. Addison-Wesley Professional, 3rd ed., 2009.

[35] T. Saito and T. Takahashi. Comprehensible Rendering of 3-D Shapes. In *Proceedings of the ACM SIGGRAPH*, pp. 197–206. ACM, New York, NY, USA, 1990.

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

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

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

[39] D. Shreiner, G. Sellers, J. M. Kessenich, and B. M. Licea-Kane. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. Addison-Wesley Professional, 8th ed., 2013.

[40] M. Stamminger and G. Drettakis. Perspective Shadow Maps. *ACM Trans. Graph.*, 21(3):557–562, July 2002.

[41] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *Proceedings of the ICCV*, pp. 839–. IEEE Computer Society, Washington, DC, USA, 1998.

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

[43] L. C. Wanger, J. A. Ferwerda, and D. P. Greenberg. Perceiving Spatial Relationships in Computer-Generated Images. *IEEE Comput. Graph. Appl.*, 12(3):44—58, May 1992.

[44] L. Williams. Casting Curved Shadows on Curved Surfaces. In *Proceedings of the ACM SIGGRAPH*, pp. 270–274. ACM, New York, NY, USA, 1978.

[45] M. Wimmer, D. Scherzer, and W. Purgathofer. Light Space Perspective Shadow Maps. In *Proceedings of the EGSR*, pp. 143–151. Eurographics Association, Aire-la-Ville, Switzerland, June 2004.

[46] A. Woo and P. Poulin. *Shadow Algorithms Data Miner*. CRC Press, Natick, MA, USA, 2012.

[47] M. W. Wright, R. Cipolla, and P. J. Giblin. Skeletonization using an extended Euclidean distance transform. *Image and Vision Computing*, 13(5):367 – 375, 1995.

[48] C. Wyman, R. Hoetzlein, and A. Lefohn. Frustum-traced Raster Shadows: Revisiting Irregular Z-buffers. In *Proceedings of the ACM I3D*, pp. 15–23. ACM, New York, NY, USA, 2015.

[49] F. Zhang, H. Sun, and O. Nyman. Parallel-Split Shadow Maps on Programmable GPUs. In H. Nguyen, ed., *GPU Gems 3*, pp. 203–237. Addison-Wesley, 2008.

[50] F. Zhang, H. Sun, L. Xu, and L. K. Lun. Parallel-split Shadow Maps for Large-scale Virtual Environments. In *Proceedings of the ACM VRCIA*, pp. 311–318. ACM, New York, NY, USA, 2006.