SCIENCE CHINA
Information Sciences

• **RESEARCH PAPER** •

November 2012 Vol. 55 No. 11: 1–12
doi: 10.1007/s11432-012-4696-2

Shadow geometry maps for alias-free shadows

WANG Rui, WU YingQing, PAN MingHao, CHEN Wei & HUA Wei*

State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310058*, China*

Received June 14, 2010; accepted March 20, 2012

Abstract Shadow maps sample scene visibility in the light source space and offer an efficient solution to generate hard shadows. However, they suffer from aliasing artifacts because of discretization errors, inadequate resolution and projection distortion. In this paper, we propose the shadow geometry map method, where a shadow depth map is augmented by storing geometry information about scenes. This leads to a new shadow-rendering algorithm that combines a supersampling filter, a geometry-aware reconstruction kernel and an irregular sampling filter. Our method produces high quality alias-free and subpixel supersampling shadow rendering and retains the simplicity and the efficiency of shadow maps. We show that the algorithm pipeline is efficiently parallelized using current programmable graphics hardware and that our method is capable of generating high quality hard shadows.

Keywords irregular shadow maps, shadow geometry maps, pixel and subpixel alias-free shadows

Citation Wang R, Wu Y Q, Pan M H, et al. Shadow geometry maps for alias-free shadows. Sci China Inf Sci, 2012, 55: 1–12, doi: 10.1007/s11432-012-4696-2

1 Introduction

Shadow is the area to which light from a light source cannot reach because of occlusion by other objects. It can be determined via computation of the light source. This can be accomplished by computing geometric representations of shadows. Because of the complex geometry of the scene and the large amount of rays needed for visibility tests, many shadow-rendering algorithms that involve geometric calculations, such as shadow volume [1] and ray tracing [2] techniques, require substantial overhead. They are less useful for real-time applications, e.g. games. This paper addresses these problems with an alias-free hard shadow-rendering algorithm.

Shadow maps [3] are a more practical approach to quick shadow generation with conventional graphics hardware. However, because of a discrete representation of the auxiliary buffer, shadow maps suffer from aliasing artifacts near the shadow boundaries. Many analysis and optimization methods have enabled recent shadow map algorithms to reduce boundary artifacts, without sacrificing performance. One kind of methods seek to optimize the sampling resolution and sample point placements of shadow maps, thereby alleviating the projection distortion [4–11]. As the approximation error caused by the discrete representation is typically unbounded, its efficiency depends on the scene configuration. The extreme of this optimization approach uses alias-free shadow maps or irregular shadow maps [12–14]. Here the

*Corresponding author (email: huawei@cad.zju.edu.cn) visibility computation is view-dependent and pixels on the shadow maps are arbitrarily placed by the irregular Z-buffer or counterparts.

Since the irregular Z-buffer does not fit naturally in the graphic pipeline, the alias-free shadow map has been impractical for real-time applications until recent hardware improvements. Recently, implementations based on CUDA [15] and Larrabee [16] have been proposed. In these methods, a two-step algorithm is used. The first step is to construct the shadow map lists, where view samples falling in the same texel are stored in one list. The second step is to rasterize triangles on the shadow map. Then visibility tests are taken between triangles and the view samples. The computational efficiency of these steps depends on the amount of view samples. For some small or slim objects, alias-free shadows at pixel level is insufficient for high visual quality. An approximated subpixel anti-aliasing method was proposed in [17]. However, for accurate computation of subpixel shadows, algorithms still require a large number of view samples. Thus we explore an more efficient algorithm for large numbers of view samples.

In this paper, we describe an alias-free and subpixel supersampling hard shadow-rendering algorithm. We base it on a thorough analysis of the causes of shadow artifacts. We propose shadow geometry maps that store depth values and geometry information about scenes. This leads to a new shadowing rendering algorithm combining a supersampling filter, a geometry-aware reconstruction kernel and an irregular projection scheme within a unified visibility resampling framework. The shadow geometry maps are related to the alias-free shadow maps used in [15,16]. But for each texel on the shadow map we store occluding triangles instead of view samples. During rendering, we first construct the shadow geometry maps and then supersample view pixels and project view samples on these maps to carry out visibility tests using geometry primitives stored in the texels.

Compared with [15,16], our algorithm has several advantages. First, by constructing lists of triangles instead of view samples on shadow maps, our algorithm is less dependent on the number of view samples. Thus it perfroms better performance when supersampled subpixel anti-aliased shadows occur. Second, as the shadow geometry maps depend only on the light sources and the scene geometry, when camera moves, it does not need to update the maps every frame. This is required by the alias-free shadow maps. Finally, as the shadow geometry maps are directly constructed from scene geometry, it is easy to integrate the silhouette map and only carry out shadow computation tests on silhouette texels.

The rest of the paper is organized as follows. First, we review related works. Next, we discusses causes of shadow aliasing by an analogy to signal processing and derive an ideal aliasing-prevention scheme. Next, we present the shadow geometry maps and the shadow-rendering algorithm. We go on to describe how we implement our technique. We then report on our results in comparison with other methods. Experiments using various examples suggest that our approach provides subpixel alias-free shadow rendering and better performance than alias-free shadow maps. Finally, we discuss the conclusions that can be drawn from this paper and options for future work.

2 Related works

Shadow volumes [1] and shadow mapping [3] are two widely used shadowing algorithms. The shadow volumes method [1] employs an auxiliary shadow volume geometry to determine the shadow boundaries. Unfortunately, problems of the robustness and high fill rates make it insufficient for real-time applications. Shadow maps [3] are more practical. However, shadow maps also have a problem, zigzag aliasing at shadow boundaries. Two types of techniques have been proposed to alleviate this aliasing. One is to use filters to smooth the aliasing and improve the visual quality. The other is to improve the match of resolutions between the shadow map and the camera space view.

Filtering techniques. Early work [4] known as percentage closer filtering filters depth maps but producing blurred shadow boundaries. Variance shadow maps [18] compute depth variances over a filter region and produce statistical approximations of the true occlusion to reduce shadow map aliasing. Convolution shadow maps [19] propose a new shadow representation that approximates shadow tests as weighted summations of basis to derive efficient arbitrary linear filtering of shadows. Aliasing artifacts are reduced by filtering and plausible results are generated. However, these methods do not produce accurate hard shadows and so do not fundamentally solve the aliasing problem.

Resolution match methods. Perspective shadow maps (PSMs) [6] try to remove perspective aliasing by applying a transform based on the camera's perspective. They can be refined using other transforms [7, 8, 20]. However, these reparameterization methods deal only with perspective aliasing. Since they apply a global warping function, complex scenes with a large depth range are not handled well. Logarithmic shadow maps [11, 21] produce lower aliasing errors but require modification to current hardware. Partitioning algorithms divide the shadow map into different parts each of which is expected to match the local sampling rate better in the camera view. The shadow map is usually partitioned according to frustum faces [9] or along the z-axis of the frustum [22]. These methods require multiple render passes hence are relatively slow. Alias-free shadow maps or irregular shadow maps [12–16] uses irregularly partitioned shadow maps to guarantee that each view sample is taken accurately in a shadow test. Our method takes similar irregular representation of alias-free or irregular shadow maps but stores different primitives.

Subpixel antialiasing shadows. Ref. [17] proposes a fast algorithm for subpixel antialiasing shadows. However, the used facet approximation makes it unable to produce accurate subpixel shadows and antialias the pixel with both shadow and shading boundaries. Although the supersampling method that rasterizes and shades more samples than at the standard resolution is slower, it is more accurate.

3 Ideal shadow maps

From a point of light in a scene, invisible regions lie in the shadow area. Such scene visibility in the context of the light is view-independent. It can be pre-computed and stored before shading. An *ideal shadow map* records every visible part of a scene in the viewpoint of the light, and is used to determine the occluded area during shading.

To study the shadow aliasing caused by conventional shadow maps, we treat the shadow determination as one-dimensional signal processing, which is shown in Figure 1. In an ideal situation (the left case in Figure 1), the scene geometry in the camera space is a continuous signal $g_c(x)$ (Figure 1(a)), and forms another continuous signal $(\boldsymbol{T}_c g_c)(x)$ (Figure 1(b)) in the light space. A shadow map can be regarded as a filter which is a finite linear combination of indicator functions of a list of intervals (Figure 1(c) and Equation 1)):

$$s_c(x) = \sum_{i=0}^{n} \alpha_i X_{I_i}(x), \text{ with } X_I(x) = \begin{cases} 1, & \text{if } x \text{ in } I, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where α_i is either 1 or 0 and I_i denotes the ith interval.

The response of the filter of a shadow map to the transformed scene geometry $(\boldsymbol{T}_c g_c)(x)$ is given by their convolution, namely, $(s_c \otimes (\boldsymbol{T}_c g_c))(x)$. Therefore, the final visible parts are piecewise continuous, and the shadow geometry indicated as the red intervals in Figure 1(d) is accurate.

Conventional shadow maps (the right hand case in Figure 1) employ discrete buffers to store the visible scene. Shadow determination is rapid through constant-time lookups and depth comparisons. The depth buffer and the shadow maps are limited in size. So the scene geometry $g_c(x)$ and the shadow map filter $s_c(x)$ are sampled into discrete signals $g(x)$ and $s(x)$ (see Figure 1(a) and (c)). Low frequencies may occur simultaneously with high frequencies, yielding heavy aliasing. Transforming the discrete signal $g(x)$ into the light space (Figure 1(b)) may enlarge the aliasing, $(\boldsymbol{T}g)(x)$, because an inverse perspective projection is involved and the degree of mismatch of the discrete signals $g(x)$ and $s(x)$ is unbounded. In extreme cases, severe zigzagged effects at very low frequencies take place in the shadow boundaries. The final shadow determination is done by a discrete filter: $(s \otimes (\boldsymbol{T}g))(x)$.

From an analysis of conventional shadow maps, aliasing in shadow maps are produced for three reasons. First, the rasterization of the shadow map $s(x)$, second, the pixel mismatch resulting from the camera-light space transformation, $(\boldsymbol{T}g)(x)$, and finally, the discrete scene geometry representation, $g(x)$. Previous works address one or two causes of this aliasing. Aliasing caused by rasterization of the shadow maps has

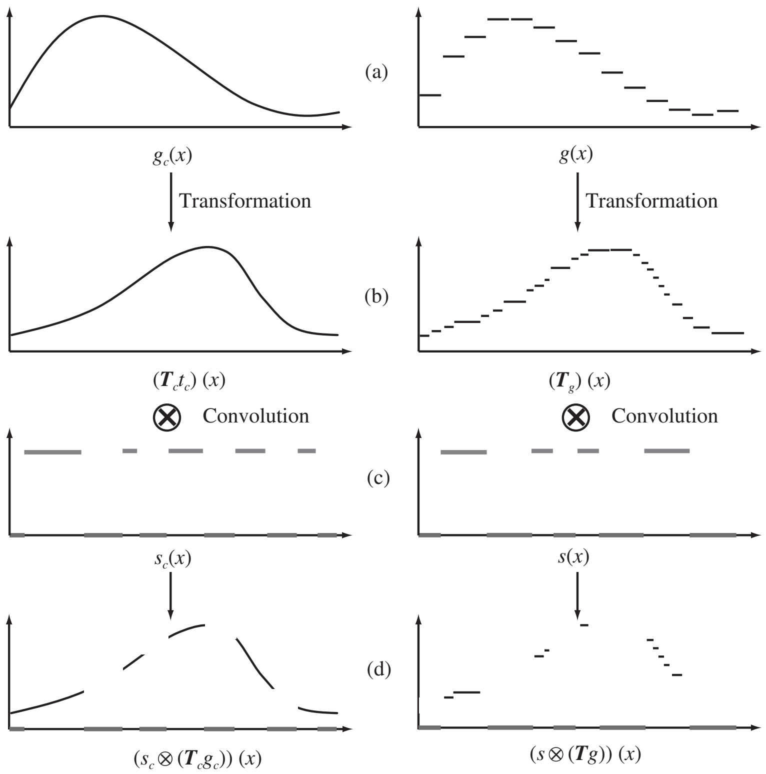


Figure 1 Ideal (left) versus conventional (right) shadow maps. (a) A continuous signal $g_c(x)$; (b) anothe continous signal $T_c g_c(x)$; (c) ashadowmap as tilter; (d) the response of the tilter of a shadow map to the transpormed scence geometry.

been addressed using shadow silhouette maps [23]. Here the shadow geometry is approximated with a piecewise constant function. Irregular Z-buffer techniques [12,14] alleviate the effects of the perspective projection transformation. However, neither method can completely avoid aliasing. In this paper, we propose a new framework to solve all of these three kinds of shadow aliasing effectively and efficiently.

4 Shadow geometry maps algorithm

To avoid these sources of aliasing and produce perfect hard shadows, we develop a new shadow rendering algorithm. Our algorithm combines a supersampling filter, a geometry-aware reconstruction kernel and a semi-regular sampling filter to generate real-time subpixel alias-free shadows. The important idea is the shadow geometry map, which stores not only the depth values but also the geometry information about scenes. In this section, we will first introduce the shadow geometry map and then describe our rendering algorithm.

4.1 Shadow geometry maps

In ideal shadow maps, one of the main sources of aliasing arises from the discrete shadow map filter $s_c(x)$. Because of the limited size of the depth buffer, shadow boundary regions with high frequencies may be sampled in low frequencies and generate zigzag aliasing. In alias-free shadow maps or irregular shadow maps [12–16], view samples are stored irregularly on the shadow map. Then an accurate shadow test is taken with scene geometry. Unlike these solutions, we preserve the scene geometry instead of view samples on shadow maps. By combing the texel-triangles list used in [17] and the 3D perspective grid used in [16], we create the geometry maps, which store additional information about the scene geometry primitives in a spatial structure, as an augmentation to shadow maps (Figure 2). The entire scene is partitioned in 3D perspective layers by several planes parallel to the shadow map plane. For each texel on the shadow map, the light frustum of the texel is divided into cells. We use the cell as an index unit

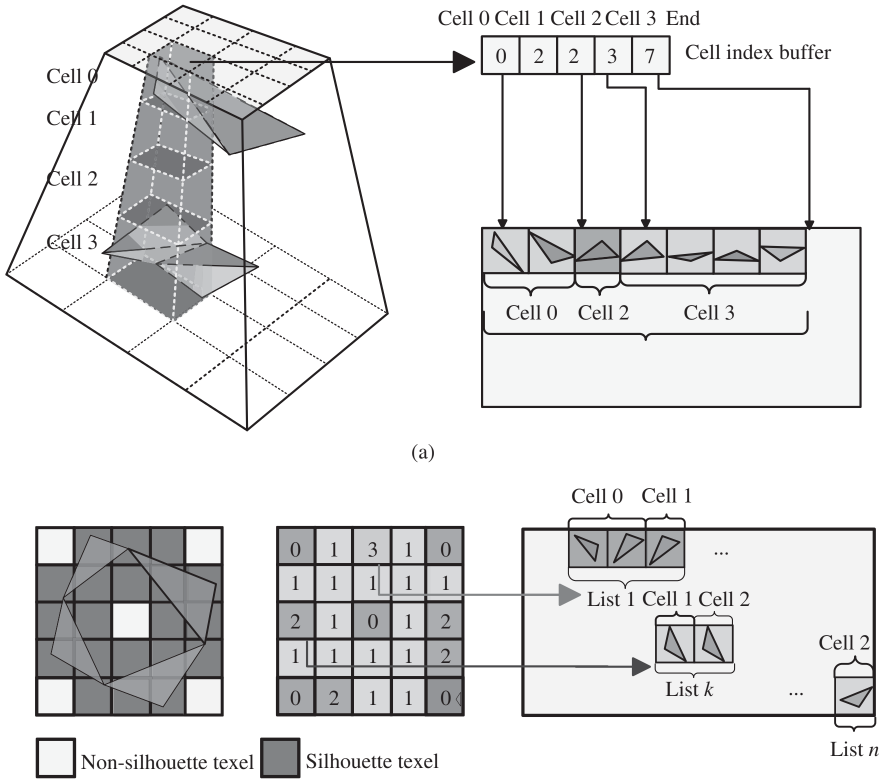


Figure 2 Illustration of the shadow geometry map and memory layout of the cell-primitives lists. (a) Illustrates the 3D perspective cells and the storage of the corresponding cell-triangles list; (b) illustrates the cell-triangles lists stored in the shadow geometry map.

to store geometry primitives. Primitives associated with each cell are stored contiguously in a 1D list, the cell-primitives list, and an extra index buffer is used to record the start and end of primitives in the list (Figure 2(a)). All cell-primitives lists are packed contiguously and stored in the order of texels in memory. Because shadow maps perform well in areas inside the shadow and aliasing artifacts only appear along shadow boundaries, we only store cell-primitives lists at shadow silhouette texels (Figure 2(b)) and depth values in non-silhouette texels.

4.2 Rendering algorithm

Our rendering algorithm combines a supersampling filter, a geometry-aware reconstruction kernel and a semi-regular sampling filter.

In computer graphics, overcoming the aliasing from the rasterization is feasible in two ways: prefiltering and supersampling. The former is performed by applying a low-pass filter to the continuous signal to get filtered signal with band-limited Nyquist frequency. But prefiltering requires sophisticated analytical filtering algorithms. It is also intractable for rasterization-based graphics pipeline. Supersampling has the effect of moving the Nyquist limit to higher frequencies by first sampling at higher resolution and then filtering down the signals. Compared with prefiltering, it is simple and easy to retrofit existing renderers. As a consequence, we apply a supersampling filter f to the rasterized scene geometry $g(x)$:

$$f \circ g(x) = \sum_{i=0}^{w} g(i) \cdot h(x-i), \tag{2}$$

where $h(x)$ is a unity mean filter with a width w and $g(i)$ denotes one scene geometry sample in the discrete sampling grid.

To reconstruct the shadow geometry, using our algorithm, we proceed to overcome the aliasing arising from the discrete depth values in the shadow map. For each view sampling point, we compute all potential occluding geometry primitives stored in shadow texels. This forms a reconstruction kernel R which ease accurate shadow geometry generation:

$$R \circ s(x) = s_c(x). \tag{3}$$

The final step computes shadows in regions that will not produce aliasing of the resolution mismatch between (a) the view sample points, (b) the shadow geometry maps and (c) the distortions of the sampling grids, when viewed from the inverse perspective projection. Visibility tests are taken irregularly based on the shadow geometry maps. The visibility of sample points lying in non-silhouette texels are evaluated rapidly by conventional shadow mapping. Visibility tests using geometry primitives stored in texels are only executed on sample points in silhouette texels. Combining these three steps results in an aliasing-free shadow map filter V, and our alias-free shadow rendering framework:

$$V(s \otimes (\boldsymbol{T}g))(x) = (R \circ s) \otimes (\boldsymbol{T}(f \circ g))(x) \tag{4}$$

$$= \sum_{i=0}^{w} s_c \otimes \boldsymbol{T}(g(i)) \cdot h(x-i). \tag{5}$$

Eq. (5) is actually an optimal reformulation of the ideal shadow maps $(s_c \otimes (\boldsymbol{T}g_c))(x)$ in the context of rasterization-based graphics hardware. Compared with the one used in conventional shadow maps $(s \otimes (\boldsymbol{T}g))(x)$, our method achieves better quality without modifying the standard rendering pipeline.

5 Implementation

In this section, we describe the implementation of our shadow rendering algorithm using the shadow geometry map in the context of one point light source. Our pipeline has two stages, constructing the shadow geometry map and rendering shadows. For brevity we suppose that the scene geometry is a collection of triangular primitives, thus the geometry primitives stored in the shadow geometry map are cell-triangles lists.

5.1 Constructing the shadow geometry maps

We follow two steps. First, for the underlying light source, we create a standard shadow map by rendering the depth values of the scene geometry in the viewpoint of the light. In addition to the depth value, each texel encodes a mask to indicate whether it is in the shadow boundaries by tracing the silhouette edges of the scene geometry. The depth of each edge whose two adjacent faces have opposite orientations is rasterized to the shadow map.

Having created the silhouette map, we know where the shadow boundary is. Storing occluding triangles requires a gathering of all scene triangles associated with the silhouette texels. We use CUDA to construct the geometry map. The implementation is similar to the CUDA implemented spatial subdivision in [24]. Triangles in the scene are processed individually to generate the cell triangle pairs. Unlike [24], only cells of silhouette texels are processed. For each cell-triangles pair, to ensure the cells of one texel are stored contiguously and in the order of the depth values, we record the texel number in the high byte and the depth value order in the low byte of the pair ID. By parallel data scanning and sorting of the pair ID, cell-primitives are then generated and stored in the geometry map.

5.2 Shadow rendering

The shadow rendering pipeline produce an accurate shadow based on our shadow geometry map. As indicated in Eq. (5), three sequential operations are required after the shadow geometry map is generated with the given light.

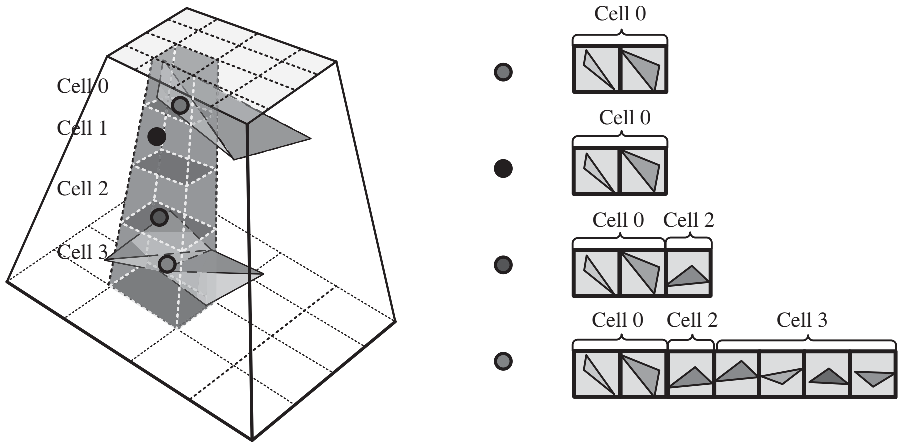


Figure 3 Visibility tests of view samples using the geometry map.

First, the depth of the scene geometry is rendered with a supersampling resolution. Sampling points $g(i)$ are used to compute the positions of the transformed sampling points $\boldsymbol{T}_c(g(i))$ in the shadow geometry maps. The shadow visibility on a sample point lying in non-silhouette texels is computed by directly comparing the depth values. For sample points lying in silhouette texels, to get $s_c \otimes \boldsymbol{T}(g(i))$, we fetch triangles stored in the geometry map and compute the occlusions. As the sample point is potentially occluded by all triangles with less depth values, we iterate over all cells with less depth values and compute the occlusion between each triangle in the cell-triangles list and the sampling point (Figure 3). These two processes account for shadow generation for low frequency (the inner or outer shadow region) and high frequency parts (the shadow boundaries). This produces an accurate visibility function $s_c \otimes \boldsymbol{T}(g(i))$. Finally, this function is convoluted with the unity mean filter

$$h(x) : \sum_{i=0}^{w} s_c \otimes \boldsymbol{T}_c(g(i)) \cdot h(x-i). \tag{6}$$

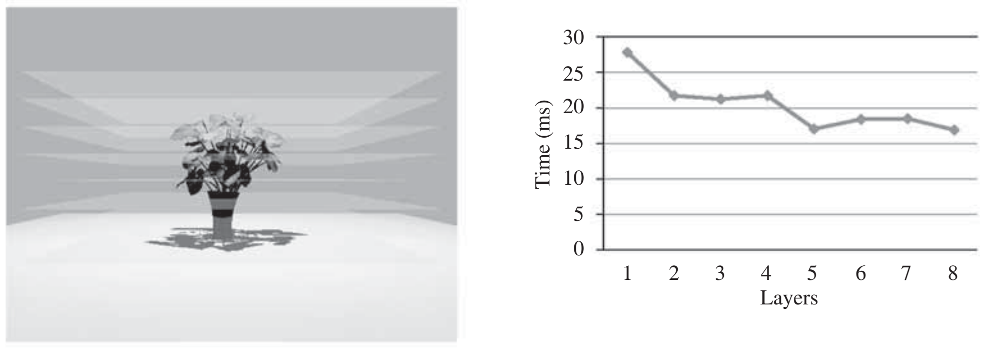
5.3 Parallelization

The advantages of using shadow geometry maps are twofold. One is the aliasing-free effects that arise from the preservation of geometry in the maps. The other is that effective speedup is favored by reducing the computational complexity of the shadow determination from a two-dimensional case (the shadow map) to a one-dimensional case (the shadow boundaries).

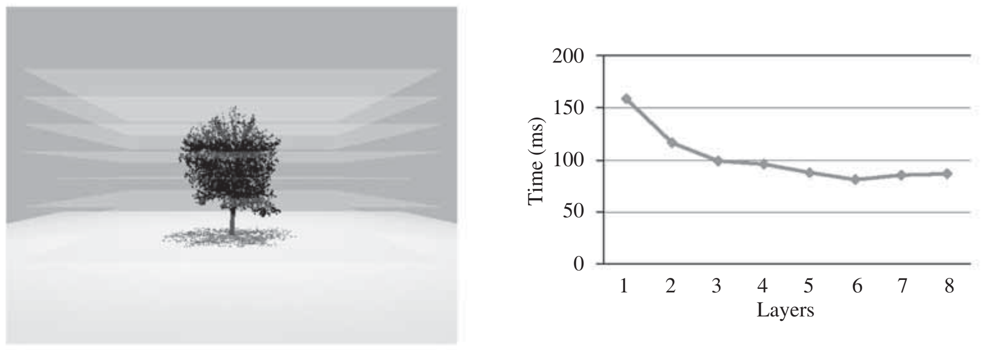
However, the geometry-preservation and sampling require multiple-pass operations and result in unbalanced algorithm complexity. Accordingly our approach addresses algorithmic parallelization: An optimized data processing granularity is designed for irregular items in the shadow geometry map. E.g., to reduce the imbalance in visibility computation between view sample points and the triangles in cell-triangles lists, we use CUDA to implement the visibility test [15,16]. We base this on the number of visibility tests of each sample point. We then reorganize the computation order of the view sample points so that in one warp (the group of parallel threads that execute concurrently in CUDA), the amount of computation in threads is similar and therefore reduce overhead arising from imbalance. Some computational resource inefficiency operations such as data scanning, data sorting and data compacting, are optimized using techniques similar to those in [10].

6 Results

We implemented our algorithm with the scalable and parallel programming language CUDA [25] and tested it on a PC with 3.0 GHz CPU, 2.0 GB host memory and an NVIDIA GeForce GTX 280 video card.



(a)



(b)

Figure 4 Illustration of two scenes and the time taken to construct different number of layers of 3D perspective grids. (a) Scene "plant"; (b) scene "tree".

First, we conducted experiments to demonstrate the accelerated construction of 3D perspective grids. Figure 4(a) shows 8 parallel layers with the light source is at the top of the scene. We tested the time to take visibility tests using the shadow geometry map with differing numbers of layers and 9x supersampling (Figure 4(b)). Our experiments gave information to guide the usage of grids in our algorithm. First, the time to construct one layer is almost constant for one scene. Second, while the number of layers increases, the total time taken for visibility tests does not always decrease. Third, the reduced time for visibility tests cannot hide the latency of construction and indexing overhead after 7 or 8 layers and the overall time taken increase. Based on these conclusions, in our following results, we use 6–8 layers.

To illustrate the shadow quality of our method, we implemented the standard shadow maps, light space perspective shadow maps and shadow silhouette maps algorithms [8, 23] with CUDA. Our results are shown in Figure 5 and for fair comparison, shadows are generated without supersampling. In all figures, the standard shadow maps, the shadow silhouette maps and the shadow geometry maps are generated at a resolution of 512×512 pixels. Figure 5 shows that standard shadow maps suffer from severe aliasing artifacts. Light space perspective shadow maps produces better results, but also do not avoid aliasing. Shadow silhouette maps get rid of aliased shadow boundaries but fail in regions with fine geometry details. In contrast, our method works well in all cases and produces the same result as the reference image, which is generated by the standard ray tracing algorithm. As our method requires extra memory beyond the standard shadow map, we also conducted memory cost comparison between PSM and our method. Because of the restriction on the resolution, we used a 4096×4096 shadow map, which occupies 64 MB of memory. By contrast, our shadow geometry map uses 30 MB of memory. The $9\times$ supersampling results are shown in Figure 6. With more memory, PSM still fails to capture fine structure of occlusions. According to our analysis in Section 3, the result meets our expectations that it cannot solve all aliasing only by increasing the resolution of shadow map. Higher resolution reduces the aliasing caused by discrete sampling and mismatch of signals. But id does not eliminate all the aliasing caused by transformation, especially in the case of shadow supersampling. Our method is usually several times slower than conventional shadow maps and related techniques. Given the quality improvement, that extra

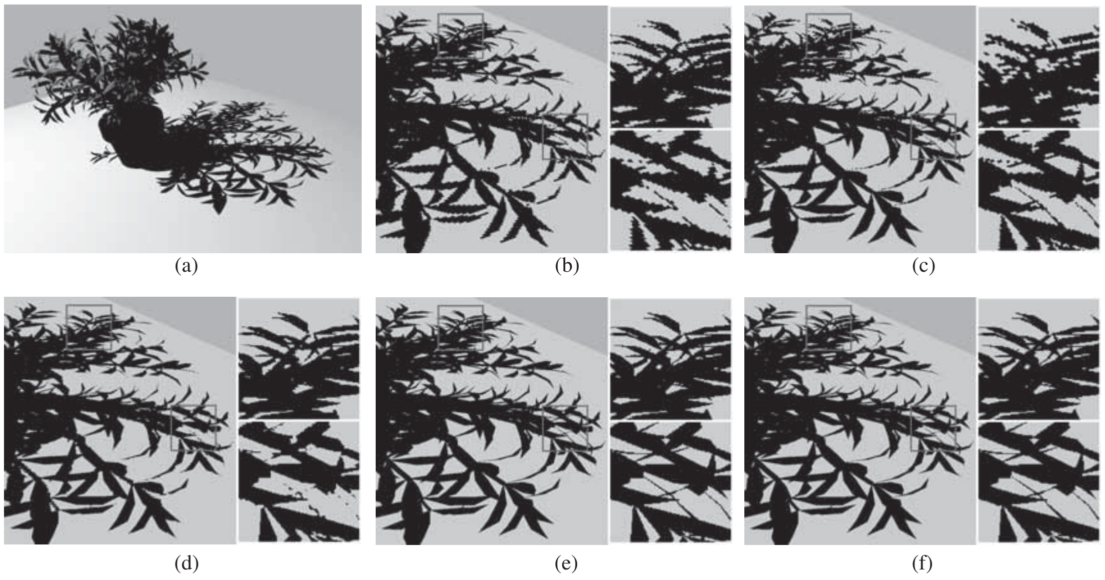


Figure 5 Quality comparison between (b) standard shadow maps, (c) light space perspective shadow maps, (d) shadow silhouette maps and (e) our result. (f) is the reference image generated by the standard ray tracing algorithm, and (a) is the original scene. (a) Scene; (b) shadow maps; (c) perspective shadow maps; (d) shadow silhouette maps; (e) ours; (f) reference.

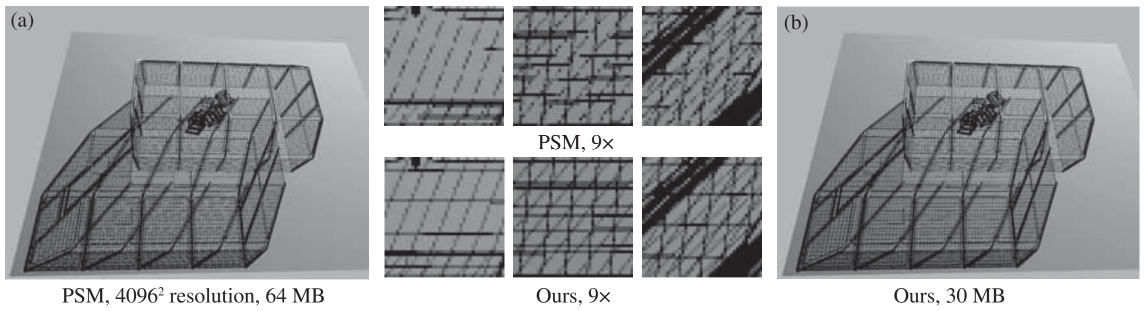


Figure 6 Quality comparison between (a) PSM and (b) our method assuming the same memory cost.

computational cost is worthwhile for applications that require high quality shadows.

In shadow geometry maps, the illumination of a scene takes two components: shading and shadow. In our implementation, both shading and shadow use the supersampling filter to overcome aliasing. The computed shadow visibility of the supersampling points in the camera space are used to modulate the shaded colors and then the colors are convoluted with the unity mean filter. Figure 7 demonstrates shadows with different supersampling resolutions, 1×, 4× and 9×. It can be observed that higher supersampling resolution provides smoother shadows hence better visual quality. Figure 8 shows the result of shadows cast by various other static objects. Figure 9 visualizes our algorithm on a large-scale urban scene. All results are presented at 9× supersampling. In these scenes, our method is capable of capturing the finest occlusions and producing shadows without artifacts. The last two scenes show the scalability of our algorithm and its practicability in wider applications. Statistics for the above scenes are summarized in Table 1. The construction of the shadow geometry maps and the frames per second (FPS) measured at different supersampling resolutions of 1×, 4× and 9× are reported. As our shadow geometry map is independent of the view point, we do not need to reconstruct the geometry map with a static light source and scene. The FPS without updating the shadow geometry map is reported in brackets. To compute the FPS, we set up a sequence of object-centered camera movements and averaged the performance observed.

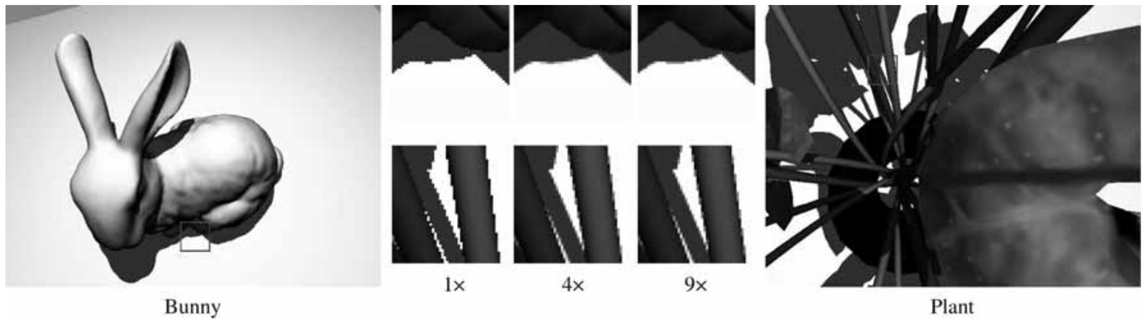


Figure 7 Shadows at different supersampling resolutions.

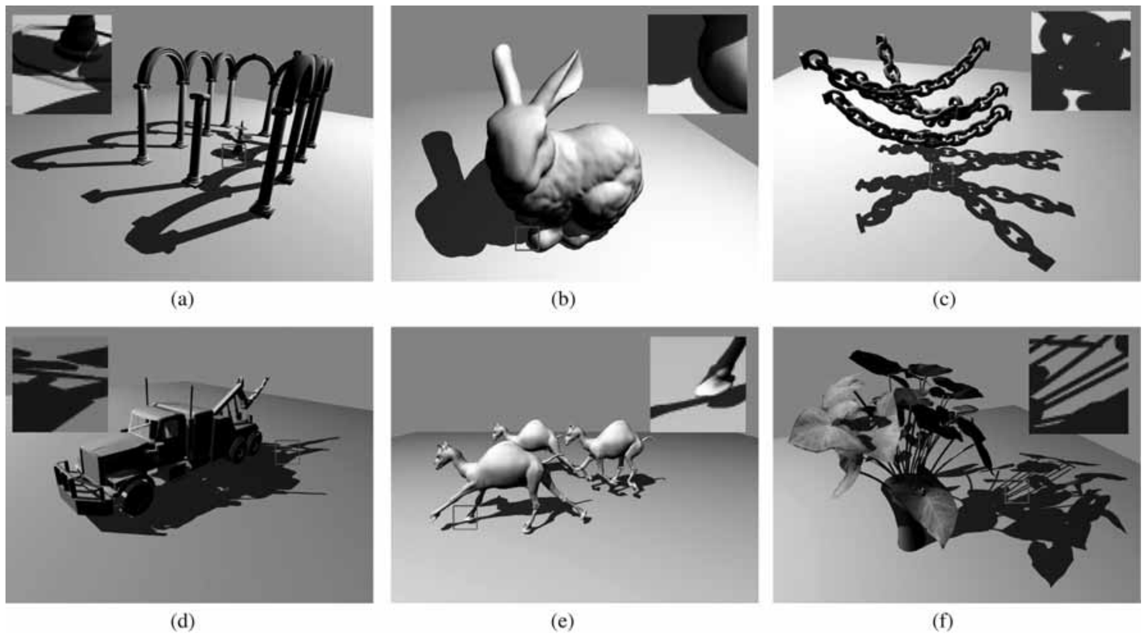


Figure 8 Our method applied to the (a) cloister, (b) bunny, (c) chain, (d) truck, (e) camels and (f) plant models.

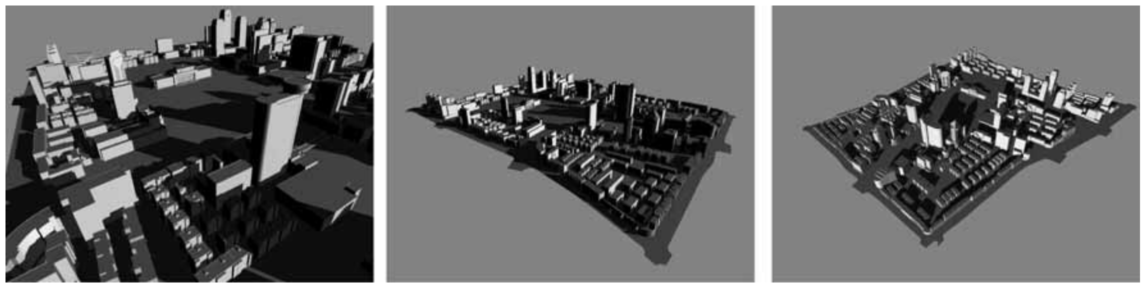


Figure 9 Shadows in an urban scene.

We also conducted experiments to compare the performance of our method with that in [15]. It also aims at producing an alias-free shadow map on a GPU. Table 1 reports the performance comparison under different supersampling resolutions. Our method not only runs faster, but it also has better scalability. While the number of view samples increases, the speedup of our method increases as well. This is mainly because our method builds cell-triangles lists instead of texel-pixels lists in [15]. Hence, in our method, the computational complexity is more associated with the scene complexity than the number of view sample points. This brings a benefit in scenes with fewer triangles. Also, we limit the shadow computation in the silhouette pixels. The method in [15] processes all screen pixels, and yields more algorithmic complexity than ours.

Table 1 Statistics of our test scenes and performance comparison with [15] under different super-sampling rates. Shadow geometry map (SGM) time is the time to construct the shadow geometry map. Frame per second (FPS) of our method is measured with and without constructing the shadow geometry map every frame. The FPS in brackets shows the case where the shadow geometry map is constructed first and then only visibility tests of view samples are computed after every following frame

Scenes	Tris.	SGM time (ms)	FPS of our method			FPS of method [15]			Speedup		
			1×	4×	9×	1×	4×	9×	1×	4×	9×
Truck	13241	2.9	140 (236)	95 (131)	64 (80)	95	44	25	1.47 (2.48)	2.15 (2.98)	2.6 (3.2)
Camels	131442	6.8	65 (116)	43 (61)	30 (38)	29	13	8	2.24 (4.02)	3.31 (4.68)	3.75 (4.75)
Bunny	69475	3.6	92 (118)	68 (95)	51 (62)	60	26	13	1.53 (1.97)	2.73 (3.67)	3.92 (4.7)
Chain	84626	6.7	69 (128)	54 (84)	37 (49)	58	26	13	1.19 (2.21)	2.08 (3.25)	2.85 (3.78)
Cloister	60596	7.2	64 (118)	43 (62)	29 (37)	45	21	11	1.42 (2.64)	2.04 (2.97)	2.63 (3.33)
Plant	34354	5.1	92 (173)	63 (92)	46 (60)	68	31	16	1.35 (2.55)	2.03 (2.99)	2.87 (3.76)
Urban Scene	212488	10.7	33 (51)	22 (29)	16 (19)	24	13	8	1.37 (2.12)	1.69 (2.23)	2.0 (2.37)

7 Conclusion and future work

Standard shadow map approaches suffer heavy aliasing artifacts because of the discrete approximation of depth values of scenes. In this paper, we have presented a versatile solution that employs shadow geometry maps to enable accurate shadow determination while taking advantage of the simplicity and flexibility of buffering techniques. The construction of shadow geometry maps and shadow rendering can be entirely implemented in programmable graphics hardware, yielding real-time performance. We have demonstrated that our approach is robust and efficient with a variety of scenes. We believe that shadow geometry maps are compatible with all state-of-the-art shadow rendering algorithms, and will work well in next-generation graphics hardware.

Our method also has some limitations. Our algorithm has benifits associated with scene complexity but this also brings a limitation. For some extremely complex scenes, our method may require more time to construct the shadow geometry map than that of building texel-pixels lists in [15]. This is the main reason that our method has a lower speedup than the method in [15] with larger scenes. It will be an interesting future work to combine our method and [15] to handle extreme complex scenes. Another limitation of our method is that it is difficult to extend it to soft shadow rendering in contrast to [15]. This will be another interesting topic to address in future.

Acknowledgements

This work was supported in part by NSFC (Grant No. 60903037), and the National Basic Research Program of China (Grant No. 2009CB320803).

References

1 Crow F C. Shadow algorithms for computer graphics. In: Proceedings of ACM SIGGRAPH. San Jose: ACM, 1977. 242–248

2 Whitted T. An improved illumination model for shaded display. Commun ACM, 1980, 23: 343–349

3 Williams L. Casting curved shadows on curved surfaces. Comput Graph, 1978, 12: 270–274

4 Reeves W T, Salesin D, Cook R L. Rendering antialiased shadows with depth maps. In: Proceedings of ACM SIGGRAPH. Anaheim: ACM, 1987. 283–291

5 Fernando R, Fernandez S, Bala K. Adaptive shadow maps. In: Proceedings of ACM SIGGRAPH. Los Angeles: ACM, 2001. 387–390

6 Stamminger M, Drettakis G. Perspective shadow maps. ACM Trans Graphic, 2002, 21: 557–562

7 Chung H, Gortler S J. A lixel for every pixel. In: Keller A, Jensen H W, eds. Proceedings of Eurographics Symposium on Rendering. Norkoping: Eurographics Association, 2004. 167–172

8 Wimmer M, Scherzer D, Purgathofer W. Light space perspective shadow maps. In: Keller A, Jensen H W, eds. Proceedings of the Eurographics Symposium on Rendering. Norkoping: Eurographics Association, 2004. 143–151

9 Lloyd B, Tuft D, Yoon S, et al. Warping and partitioning for low error shadow maps. In: Akenine-Muller T, Heidrich W, eds. Proceedings of the Eurographics Symposium on Rendering. Nicosia: Eurographics Association, 2006. 215–226

10 Lefohn A E, Sengupta S, Owens J D. Resolution-matched shadow maps. ACM Trans Graphic, 2007, 26: 1–17

11 Lloyd B, Govindaraju N, Quammen C, et al. Logarithmic perspective shadow maps. ACM Trans Graphic, 2008, 27: 1–106

12 Aila T, Laine S. Alias-free shadow maps. J Graphic Tool, 2007, 12: 47–59

13 Johnson G, Mark W, Burns C. The irregular Z-Buffer and its Application to Shadow Mapping. Technical Report TR-04-09. Austin: University of Texas, 2004

14 Johnson G S, Lee J, Burns C A, et al. The Irregular Z-buffer: hardware acceleration for irregular data structures. ACM Trans Graphic, 2005, 24: 1462–1482

15 Sintorn E, Eisemann E, Assarsson U. Sample based visibility for soft shadows using alias-free shadow maps. Comput Graph Forum, 2008, 27: 1285–1292

16 Johnson G S, Hunt W A, Hux A, et al. Soft irregular shadow mapping: fast, high-quality, and robust soft shadows. In: I3D '09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games. Boston: ACM, 2009. 57–66

17 Pan M H, Wang R, Chen W F. Fast, sub-pixel antialiased shadow maps. Comput Graph Forum, 2009, 12: 1–34

18 Donnelly W, Lauritzen A. Variance shadow maps. In: I3D '06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games. Redwood City: ACM, 2006. 161–165

19 Thomas A, Tom M, Hans-Peter S, et al. Convolution shadow maps. In: Kautz J, Pattanaik S N, eds. Rendering Techniques 2007: Eurographics Symposium on Rendering. Grenoble: Eurographics Association, 2007. 51–60

20 Martin T, Tan T S. Anti-aliasing and continuity with trapezoidal shadow maps. In: Keller A, Jensen H W, eds. Rendering Techniques 2004: Proceedings of Eurographics Symposium on Rendering. Norkoping: Eurographics Association, 2004. 51–60

21 Lloyd B, Govindaraju N K, Molnar S E, et al. Practical logarithmic rasterization for low-error shadow maps. In: Stephen N, ed. Proceedings of ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware. Switzerland: Eurographics Association, 2007. 17–24

22 Zhang F, Sun H Q, Xu L L, et al. Parallel-split shadow maps for large-scale virtual environments. In: Sun H Q, ed. Proceedings of ACM International Conference on Virtual Reality Continuum and its Applications. New York: ACM, 2006. 311–318

23 Sen P, Cammarano M, Hanrahan P. Shadow silhouette maps. ACM Trans Graphic, 2003, 22: 521–526

24 Grand S L. Broad-phase collision detection with CUDA. In: Nguyen H, ed. GPU Gems 3. Addison-Wesley, 2007. 697–721

25 Nickolls J, Buck I, Garland M, et al. Scalable parallel programming with CUDA. In: ACM SIGGRAPH 2008 Classes. Los Angeles: ACM, 2008. 1–14