Multiple-Depth Shadow Maps

Christian Azambuja Pagot João Luiz Dihl Comba Manuel Menezes de Oliveira Neto
Instituto de Informática - UFRGS
{capagot, comba, oliveira}@inf.ufrgs.br

Abstract

Traditional shadow maps store a single depth value per cell, leading to a binary outcome by the shadow test (either lit or in shadow), and are prone to produce aliased shadow borders. We present a new approach that produces better estimates of shadow percentages and, in combination with percentage closer filtering (PCF), reduces aliasing artifacts using smaller kernel sizes. The new algorithm extends the notions of shadow maps and shadow tests to support the representation of multiple depth values per shadow map cell, as well as multi-valued shadow tests. This new approach has the potential for hardware implementation, but can also be implemented exploiting the programmable capabilities of recent graphics cards.

1. Introduction

Shadows are very important scene elements, providing visual cues about the spatial relationships among objects and light sources in a scene. Recently several algorithms are being revisited for possible hardware implementation. In the case of shadow computation, the Shadow Mapping algorithm originally proposed by Williams [21] is suitable for such an implementation due to its simplicity and generality. However, it suffers from aliasing artifacts and self-shadowing, problems still under investigation [13].

In this work, we present an extension of Shadow Mapping called *Multiple-Depth Shadow Mapping* (MDSM), that significantly reduces aliasing artifacts with a small increase in computational cost. MDSM supports the representation and test of multiple depth values per shadow map cell. As a result, the algorithm produces better estimates of shadow percentages using the Percentage Closer Filtering (PCF) algorithm. We demonstrate the advantages and effectiveness of this new approach both analytically and experimentally.

Figure 1 illustrates the proposed technique in a sample scene comparing details of its shadows in different situations. In Figure 1(b) the shadow boundary is obtained with the use of conventional shadow mapping in combination with a 4x4 PCF kernel. Figure 1(c)-(d) shows the results obtained using a 3x3 PCF kernel and the MDSM approach with 2 and 3 depth values respectively. Note that the transitions between the lit and shadow regions are similar, showing that the use of smaller PCF kernels with MDSM solutions produce results comparable to the standard shadow mapping solution with bigger PCF kernels.

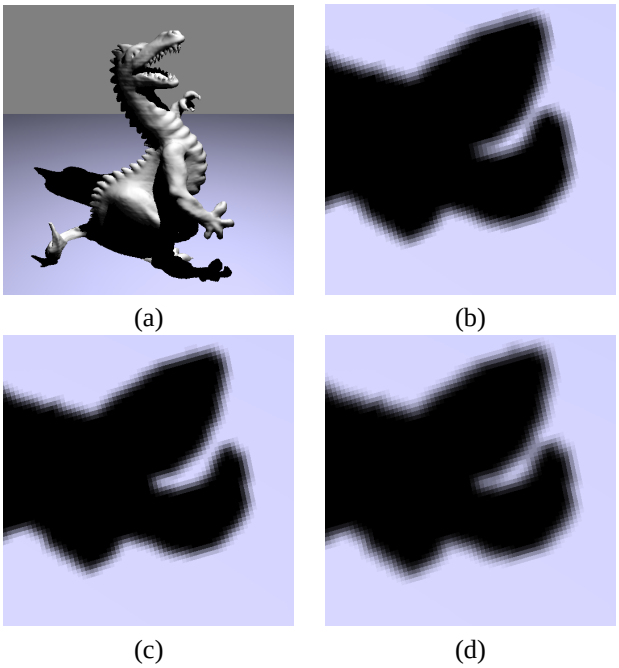

Figure 1. (a) Sample scene containing shadows. (b) Shadow borders computed using the conventional shadow mapping algorithm with a 4x4 PCF kernel. Same view using a (c) two-depth shadow map with a 3x3 PCF kernel and (d) three-depth shadow map with a 3x3 PCF kernel.

The main contributions of this work can be summarized as follows

- A generalization of the notions of shadow map and shadow test to support the representation of multiple depth values per shadow map cell. The new multiple-depth shadow test can produce a larger range of shade outcomes when compared to the conventional shadow test (Section 4).
- An analytical characterization of the quality of the shadows created by the Shadow Mapping algorithm as a function of the number of different values produced by the depth test, and the number of samples used by the PCF algorithm (Section 4).
- An extension of the Shadow Mapping algorithm capable of producing good quality antialiased shadow boundaries through the use of smaller PCF kernels. (Section 4).

The remaining of the paper is organized as follows: Section 2 provides a review of the original Shadow Mapping [21] and PCF [15] algorithms, followed by a discussion of related work in section 3. Section 4 formalizes the notions of multiple-depth shadow maps and presents a particular formalization of multiple-depth shadow tests. It also discusses some design alternatives for the implementation of multiple-depth shadow maps. Section 5 presents some results obtained with the proposed approach, followed by a discussion on the use of MDSMs. Section 7 summarizes the paper.

2. Background

2.1. The Shadow Mapping Algorithm

Several algorithms for shadow computation have been described in the literature, and good surveys can be found in [3], [24], [13] and [6]. Shadow algorithms are usually classified according to the space in which shadow computation takes place: object-space or image-space. *Object-space* algorithms involve geometric computation of the shadow, such as in shadow-volume computations [8], [7], and ray-tracing [20]. By contrast, *image-space* algorithms compute shadows based on visibility information obtained in image representations. The Shadow Mapping algorithm [21], also called backward Shadow Mapping, is the classical example of this class and constitutes the focus of this paper.

The Shadow Mapping algorithm [21] is a two-pass technique for shadow generation. In the first pass, the scene is rendered from the point of view of the light source into a depth image (shadow map). In a subsequent step, the scene is rendered again, this time from the camera's viewpoint. In order to decide whether each pixel in the camera's view is in shadow with respect to the light source, the coordinates of the point are transformed from eye space into light space, and projected into the light source image plane. We access the shadow map to recover the corresponding depth value, and if the stored value is smaller than the projected depth, the point is considered in shadow; otherwise it is lit.

This algorithm works for any shape that can be rendered into a z-buffer. Unfortunately, it suffers from aliasing and limited precision of depth values. Aliasing happens because a shadow map is a discrete representation of visibility information considered from the light source viewpoint. Limited precision can lead to self-shadowing artifacts.

2.2. Percentage Closer Filtering

The Percentage Closer Filtering algorithm [15] (PCF) significantly reduces aliasing artifacts in shadow mapping applications. It performs a filtering operation over the outcomes of shadow tests (0 or 1) produced by the Shadow Mapping algorithm. For each cell, PCF performs shadow tests at locations defined by a regular grid or using stochastic sampling. The outcomes from all shadow tests are averaged to produce the percentage of pixels in shadow. Figure 2 shows an example using a 3x3 PCF kernel.

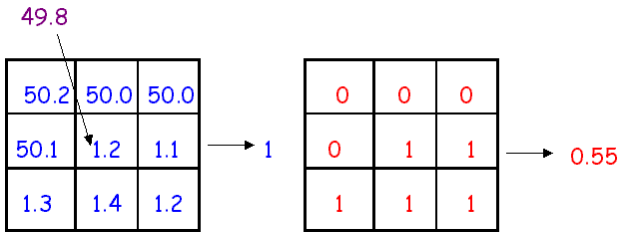

Figure 2. In the PCF algorithm, additional shadow tests are performed to compare a given depth value (49.8) against depth values of neighboring elements in a shadow map. The resulting outcomes are combined to determine the amount of attenuation to be applied to the pixel (5/9=0.55).

PCF became popular due to its implementation in Renderman [17], a rendering system targeted towards off-line rendering. While the results obtained with the use of PCF can be quite impressive, the big disadvantage of the method is its cost: for reasonable results, it is necessary the use of filters with 16 to 25 samples, which substantially increases the number of shadow map accesses and computations per pixel. This makes the use of PCF less attractive for interactive applications.

3. Related Work

The PCF approach [15] addresses the aliasing artifacts inherent to Shadow Mapping. Although the biggest cause of aliasing in the shadow mapping algorithm is the use of inadequate shadow map resolution, several approaches perform antialising by simply blurring shadow boundaries. Fernando et al. [10] minimizes aliasing artifacts by replacing ordinary shadow maps with an adaptive hierarchical structure that provides higher resolution to regions containing shadow boundaries.

The first dedicated shadow mapping hardware appeared in the high-end Reality Engine [2] by SGI. Before dedicated shadow-mapping hardware was available in the commodity graphics cards, the shadow test was simulated using the alpha test on the resulting texture values. Heidrich [12] extended the precision of the depth test from 8 to 16 bits using a combination of two 8-bit alpha tests that can be performed in the pixel shader hardware of NVIDIA's GeForce series. The advance of graphics hardware allowed shadow maps to be implemented in hardware using the concept of projective textures [16]. Such dedicated hardware [13] can be found on consumer graphics boards from NVIDIA and ATI.

Although the use of multiple depth values in shadow maps has also appeared in some other papers [19] [23] [18], the way they were used, and intended purposes, were different from ours. Layered Attenuation Maps [1] use an image-based rendering technique to simulate soft shadows. Deep shadow maps [14] provides an efficient way to compute realistic shadows for structures such as hair, fur and smoke. Unlike conventional shadow maps, deep shadow maps store a visibility function at each cell. Such functions represent how much light is attenuated as it penetrates the object. Aliasing reduction is addressed with the use of jittered samples and mip-mapping [22].

Brabec and Seidel [5] present a hardware implementation of PCF called Fast PCF. It implements a PCF with 4 samples by representing 4 depth values (8 bits each) in a single texture element. We generalize the concepts exposed in [5], further exploring how the PCF can benefit from multiple depth values.

4. Multiple-Depth Shadow Maps

4.1. Motivation

The motivation for this work was to reduce the cost of using PCF without degrading the quality of shadow boundaries. For this discussion, we need first to evaluate the costs of the shadow map computations.

The overall cost of a traditional shadow mapping computation is related to: (1) the number of fragments generated by the scene (called f), (2) the transformation from world to light space (called $cost_{transf}$), (3) a texture access to the shadow map to recover a depth value (called $cost_{texture}$) and (4) the shadow test (called $cost_{st}$). The use of PCF requires issuing several shadow tests and averaging their results into a quantity that express the percentage of shadow associated with central pixel. If a PCF with a kernel of k samples is used, the shadow test and texture costs are scaled by k. The overall cost of the shadow mapping algorithm can be expressed as:

$$cost_{sm} = f \times (cost_{transf} + k \times (cost_{texture} + cost_{st})) \quad (1)$$

The distinct results (shadow percentages) returned by a PCF kernel with k samples is $k + 1$. Increasing k improves PCF results, allowing more distinct shadow percentages at the cost of additional texture accesses. On the other hand, if the results of the shadow test were not constrained to a binary range, the number of shading values generated by the PCF filter would increase at the expense of only additional logical tests. One way to accomplish this is through the use of a more general shadow map, capable of storing more then one depth value per cell.

4.2. Generalization of Multiple-Depth Shadow Map

In this section we formalize the notions of Multiple-Depth Shadow Map and Multiple-Depth Shadow Test.

- **Multiple-Depth Shadow Maps**: A *multiple-depth shadow map* (n-SM) is a shadow map that stores n depth values per cell. For convenience, we will assume that inside each cell the n depth values, $\{Z_1, Z_2, ..., Z_n\}$, are sorted in descending order, i.e., $Z_1 \geq Z_2 \geq ... \geq Z_{n-1} \geq Z_n$.
- **Multiple-Depth Shadow Test**: An *n-shadow test* (n-ST) is an extension of the regular shadow test that performs up to n comparisons against the ordered values stored in a shadow map cell. The outcome of the test is given by the ratio $sc = s/n$, where $s \in \{0, .., n\}$, and can assume $w = n+1$ different and equally spaced values in the range $[0, 1]$. Conceptually, the process can be described as follows: given a depth value d to be tested against an n-SM, the shadow coverage returned by the n-ST is $sc = 1$, if $d > Z_n$, otherwise $sc = (i - 1)/n$, where Z_i is the largest Z value (stored in the n-SM) smaller than d. A traditional Shadow Map is a special case of a Multiple-Depth Shadow Map for which $n = 1$, and can be written as 1-SM. Likewise, its corresponding depth test is a 1-shadow test (1-ST).

A PCF can be applied to a multiple-depth shadow map in the regular way. In order to make our description more precise, the following notation is introduced.

- **PCF(k,n)**: A PCF(k,n) is a k-sample PCF applied to a n-SM. It can produce up to $d_{max} = nk + 1$ different outcomes. The value d_{max} is obtained by recalling that an n-depth test can produce up to $w = n + 1$ different results (*i.e.*, up to n non-zero values), adding up to a total of $d_{max} = nk + 1$ different outcomes.

Using this notation, a PCF(1,1) is simply the conventional shadow map proposal [21]. A PCF(4,1) is a PCF using 4 samples applied to a conventional shadow map, which produces up to 5 different outcomes (0, 0.25, 0.5, 0.75, 1).

Table 1 shows the number of outcomes produced by the PCF(k,n) for some values of k and n.

$k \setminus n$	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9
4	5	9	13	17	21	25	29	33
9	10	19	28	37	46	55	64	73
16	17	33	49	65	81	97	113	129
25	26	51	76	101	126	151	176	201

Table 1. Number of outcomes produced by a PCF(k,n),

From table 1 one sees that a PCF(9,2) can produce up to 19 ($2 \times 9 + 1$) different shading values, whereas the maximum number of shading values produced by a PCF(16,1) is 17 ($1 \times 16 + 1$). In other words, the use of a 2-SM with a 9 sample PCF can potentially produce smoother shadow borders than a conventional shadow map (1-SM) with a 16 sample PCF. For this particular example, one can expect an improvement in shading quality of 11.7% using only 56% of the PCF kernel samples. Figure 3 compares the number of shading values produced by MDSMs (for values of n varying from 1 to 4 when used in combination with PCFs of various sizes.)

4.3. Selecting Depth Values

In this section we focus our discussion to the selection of depth values for two specific classes of MDSMs: the 2-SM and 3-SM. Selecting depth values for higher-order shadow maps is more involved and still open for investigation.

For each cell of a MDSM, the selection of depth values must always include the corresponding depth value of the standard shadow map. This is required to preserve the basic structure of the shadow. Therefore, only one additional depth value is required for the 2-SM, and two for the 3-SM.

One of the goals of using additional depth values is to increase the quality of shadow boundaries, which can be accomplished by increasing the number of outcomes produced by the shadow test. Therefore, the choice of additional depth values is critical for cells that are closer to shadow borders. The information encoded in the standard shadow map is used for this purpose, which can represent sharp transitions along depth discontinuities, flat regions of constant depth or smoothly varying depth regions.

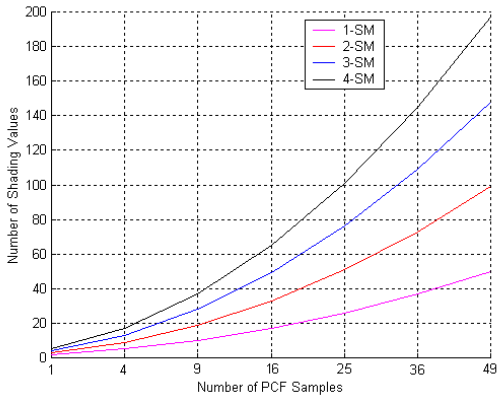

Figure 3. Number of shading values obtained by the combination of Multiple-Depth Shadow Maps and PCF with different number of samples.

We select additional depth values that make discontinuities more evident by evaluating q samples from within a search region of the base shadow map. Once all samples are collected, we proceed to choose the best representatives (one or two depth values for the 2-SM and 3-SM). Two simple ways to do that include:

- *Sorting*: Sort all sample values and divide them into n groups, picking a representative from each one. This requires a $O(q\ logq)$ effort.
- *Maximum* or *minimum* value: Chooses the maximum or minimum (or both) values, requiring a $O(q)$ effort.

Since the search must be performed efficiently, a small q and a simple search criteria must be used. Therefore, the maximum or minimum represent the cheapest solution (since it is not necessary to sort the values). Figure 4 shows the use of the composed shadow test in a 2-SM. Depending on the values passed to the 2-ST test, the shading percentages that could be returned are: 0.0 (fully lit), 0.5 (50% shadowed) and 1.0 (completely in shadow). The figure presents a situation where the pixel is 50% shadowed.

4.4. Hardware Implementation

MDSMs have a good potential for hardware implementation. In the case of static scenes and light sources, the MDSM can be pre-computed and therefore the cost of

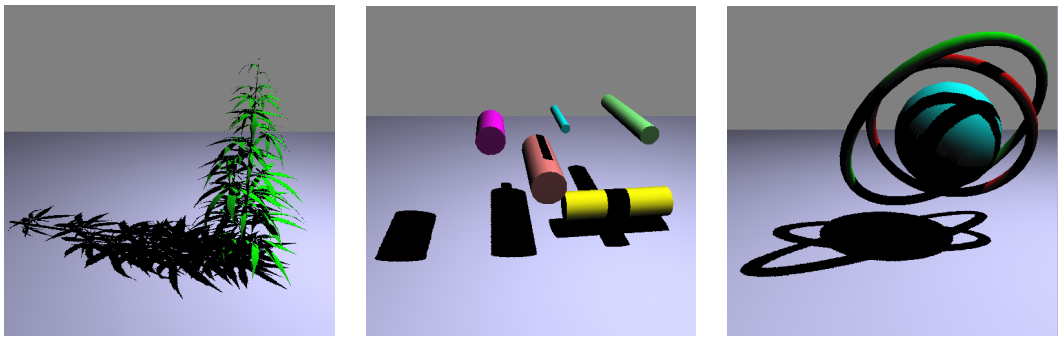

Figure 5. Additional scenes used to test our algorithm

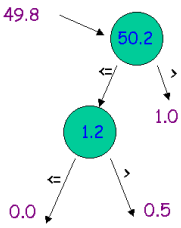

Figure 4. Instance of a 2-SM decision tree.

choosing additional depth values does not need to be considered. For dynamic scenes, it would be advantageous having a special hardware to select additional depth values, (e.g. the minimum or maximum of a neighborhood around each shadow map cell).

Another possibility is to consider using occlusion culling hardware of recent graphics hardware. The ATI's Hyper-Z technology is capable of selecting the minimum depth value among four or sixteen depth values. However, such samples are taken from a tiled subdivision of the image plane, which leads to severe shading artifacts at nearly regular intervals. Changes to this hardware might allow for a faster and proper selection of additional depth values.

Although a faster selection of the additional depth values implies in some hardware modification, the storage of a 2-SM or 3-SM does not require any modification. The texture formats found on actual graphics hardware allows for up to 4 readable/writable channels, with 32-bit floating point precision each. It means that a 4-SM could be stored in each cell of such a texture, and that the 4 depth values could be efficiently fetched with only one texture access.

5. Results

We have implemented the 2-SM and 3-SM algorithms exploiting the programmable shading capabilities available on the NVIDIA GeForce 5800. Code was written using OpenGl, Cg and C++. The 2-SM (3-SM) was implemented as a 2D texture, with the z values stored in different 32-bit floating-point color channels. This greatly improves the overall performance since the z values can be fetched with only one texture access. In order to compare the results of our method with the combined use of conventional Shadow Mapping and PCF, we have implemented PCF kernels using 3x3 (=9 samples) and 4x4 (=16 samples). The second and third depth values were chosen as the maximum and minimum values on a neighborhood in the first shadow map. Scene and shadow maps were computed at the resolution of 512^2.

Figure 5 shows sample scenes used to evaluate the conventional and MDSM algorithms. Figure 6 illustrates a close view of the shadows projected on a planar surface, rendered using different combinations of number of depth values and number of PCF samples. In this example we illustrate that results obtained with a standard shadow map using a PCF kernels of 3x3 and 4x4 are comparable to the results using 3-SM and a PCF kernel one dimension smaller (2x2 and 3x3 respectively).

In Figure 7 we illustrate shadows casted into curved surfaces, and compare the use of the standard shadow mapping algorithm and a 4x4 PCF filter against a 2-SM and 3-SM using 3x3 PCF kernels. As in the previous example, the results are very similar. Note that the 3-SM produces a transition between lit and shadow regions closer to the one generated in the standard shadow map (same transition using the 2-SM is thinner).

6. Discussion

The advantages of MDSMs come at the expense of selecting and storing multiple depth values in the shadow map, and executing a more complex shadow test. At a first glance, it might appear that we have just transferred some overhead from the PCF filtering phase to the construction of the MDSM. A careful analysis shows, however, that,

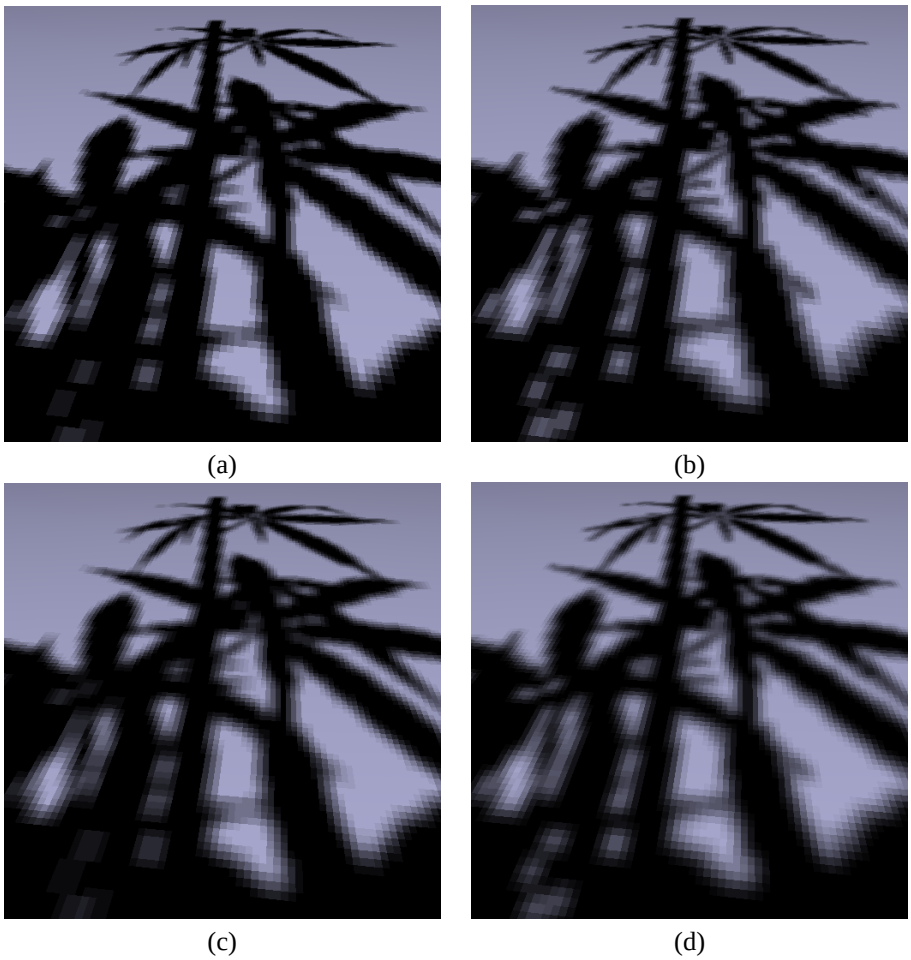

Figure 6. Comparison of standard shadow maps and MDSMs. (a)(b) standard shadow map using PCF 3x3 and PCF 4x4 kernels respectively. (c)(d) 3-SM using a PCF 2x2 and PCF 3x3 kernels respectively. Note how similar the results with smaller kernels using MDSM are to standard shadow maps (compare (a) against (c) and (b) against (d)).

for special situations, the costs associated with MDSM are smaller than with the standard shadow map.

A multiple-depth shadow test is slightly more expensive than a conventional shadow test, but it is reasonable to assume that logical tests are cheap and dedicated hardware can perform all comparisons in parallel. The overhead generated during the selection of additional depth values is of major concern. Our algorithm breaks the problem of filtering in two parts, where the cost of one part is fixed and known, which does not happen with the traditional PCF algorithm.

An example helps illustrates this point, and focus our discussion on a 2-SM (cost analysis for a 3-SM is similar). Suppose one uses a maximum (or minimum) strategy for choosing additional depth values in a q-sample area around each cell. It implies that q texture fetches will be executed around each cell of the shadow map. Let h_{sm} and w_{sm} be the dimensions of the shadow map. The cost to generate all the second values of the 2-SM ($cost_{2value}$), disregarding logical tests and latency times, is given by:

$$cost_{2value} = q \times h_{sm} \times w_{sm} \tag{2}$$

Also, let f be the number of fragments generated by the rasterization of the scene from the camera's view point and let k be the number of samples of a PCF kernel, the cost to generate a 1-SM antialiased shadowed scene ($cost_{scene}$) can be expressed by

$$cost_{scene} = f \times k \tag{3}$$

The cost of a scene generated with a 2-SM or 3-SM can be expressed as the sum of the cost to select addi-

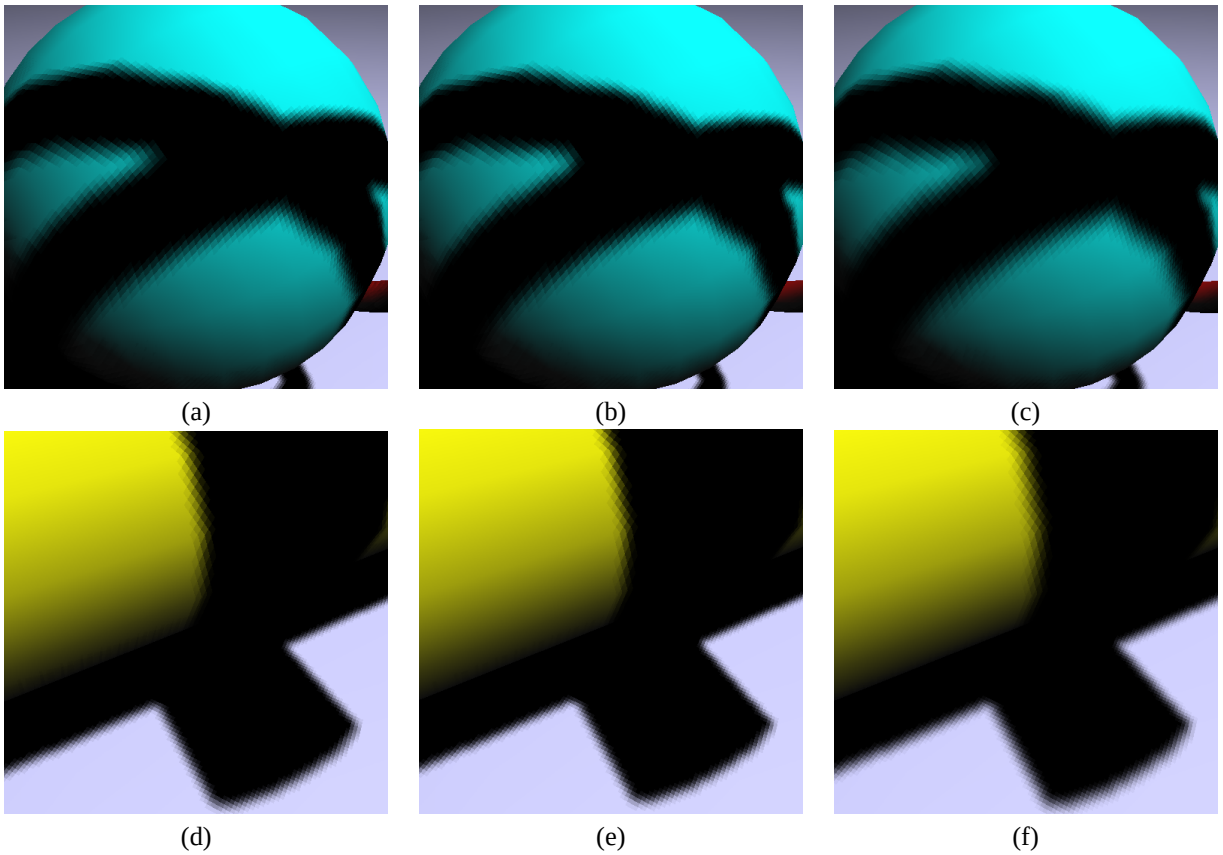

Figure 7. Comparison of standard shadow maps and MDSMs in non-planar shadow boundaries. (a)(d) standard shadow map and PCF 4x4. (b)(e) 2-SM and PCF 3x3. (c)(f) 3-SM and PCF 3x3.

tional depth values and the cost to generate the final antialiased shadowed scene:

$$cost_{2sm\_scene} = q \times h_{sm} \times w_{sm} + f \times k \qquad (4)$$

The number of fragments (f) generated by an arbitrary scene is variable and difficult to predict. If the scene presents high depth complexity, the number of fragments it can produce can be several times bigger than the screen resolution. On the other hand, the number of fragments generated by the shadow map, during the generation of its second values, is fixed and equal to its resolution. Thus one can conclude that, as the number of fragments generated by a scene grows, there will be a point where the cost to generate a PCF($k + r$,1), where $r \in \mathbb{N}^*$, for a scene will be greater than the cost to generate a PCF(k,2), and it can be expressed as the point where

$$f > \frac{q \times h_{sm} \times w_{sm}}{r} \qquad (5)$$

For example, consider the resolutions used in our tests (512^2 for both screen and shadow maps), and compare the cost of the standard shadow map using a PCF with 16 samples (4x4) against the 2-SM using a PCF with 9 samples (3x3) - r is equal to $16 - 9 = 7$. In this case, the MDSM has a smaller cost when $f > 1.28 \times 512^2$. This represents slightly more fragments than screen resolution, and happens frequently for scenes with moderately depth complexity.

One can argue that visibility calculations can be computed before shadowing takes place (deferred shading [9]). In this case, we assume that shadow computation is executed only once for each screen pixel ($f = h_s \times w_s$), where h_s and w_s represent the screen dimensions. If we express $h_s = \lambda_h * h_{sm}$ and $w_s = \lambda_w * w_{sm}$ as scaled values of shadow map dimensions, the use of a 2-SM is cheaper than conventional shadow maps with bigger PCF kernels when:

$$\lambda_h \times \lambda_w > \frac{q}{r} \qquad (6)$$

If the screen and the shadow maps have the same dimensions, this inequality will never be satisfied if q is larger than r (which is the case in the above example). However,

is not uncommon to have shadow maps with smaller resolution than screen (some computer games use small shadow maps to locally project shadows on characters). For the case above (shadow map resolution equals to 512^2), it suffices the screen resolution to be 1.28 larger than that (a 656^2 resolution) for the 2-SM have a smaller cost.

7. Conclusions and Future Work

We introduced the notion of MDSMs as a generalization of conventional shadow maps that support the representation of multiple depth values per shadow map cell as well as multi-valued shadow tests. When combined with PCF, this new structure produces good estimates of shadow percentages and smoother shadow boundaries. Compared with conventional Shadow Mapping, a smaller number of PCF samples are needed to obtain renderings of comparable quality. We have demonstrated the effectiveness and advantages of our approach both analytically and experimentally. Given that Shadow Mapping has been traditionally implemented using texture mapping hardware, our approach can significantly reduce the number of texture accesses to shadow maps required at each frame. This is specially important since texture accesses are a limiting factor preventing current raster architectures from achieving higher frame rates.

MDSMs have good potential for hardware implementation and should require only minor changes to the current graphics pipeline. Assuming that a depth value can be satisfactorily represented using 16 bits, 128-bit texture memories already available on recent graphics accelerators can support up to 8 depth values per cell (8-SM). Alternatively, MDSM can be implemented exploiting the recent programmable capabilities available on current graphics cards. As a future work, implementations directed towards performance evalutation will be made. We also intend to test different criteria for the selection of additional depth values.

References

[1] M. Agrawala, R. Ramamoorthi, A. Heirich (HP) and Laurent Moll (HP) Efficient Image-Based Methods for Rendering Soft Shadows *Siggraph 2000 Conference Proceedings*, pp. 375-384. Addison Wesley, August 2000.

[2] K. Akeley. RealityEngine graphics. In *SIGGRAPH 1993 Conference Proceedings*, Annual Conference Series, August 1993, pp. 109–116.

[3] T. Akenine-Moller and E. Haines. Real-Time Rendering. 2nd edition. A K Peters, 2002.

[4] ATI Technologies Inc. Performance Optimization Techniques for ATI Graphics Hardware with DirectX 9.0. In http://www.ati.com/developer/dx9/ATI-DX9_Optimization.pdf.

[5] S. Brabec, Hans-Peter Seidel. Hardware-accelerated Rendering of Antialiased Shadows with Shadow Maps. In Computer Graphics International (CGI) 2001 proceedings

[6] S. Brabec, T. Annen, and Hans-Peter Seidel. Practical Shadow Mapping. In *Journal of Graphics Tools*, 7(4):9-18, 2002.

[7] Chin, N., Feiner, S. Near Real-Time Shadow Generation Using BSP Trees. In *SIGGRAPH 1989 Conference Proceedings*, Annual Conference Series, July 1989, pp. 99-106.

[8] F. C. Crow. Shadow Algorithms for Computer Graphics. In *SIGGRAPH 1977 Conference Proceedings*, Annual Conference Series, July 1977.

[9] Deering, M., S. Winner, B. Schediwy, C. Duffy, and N. Hunt. The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics. In *SIGGRAPH 1988 Conference Proceedings*, pp. 21-30, 1988.

[10] R. Fernando, S. Fernandez, K. Bala, D. P. Greenberg. Adaptive Shadow Maps. In *SIGGRAPH 2001 Conference Proceedings*, pp.387-390. Addison Wesley, August 2001.

[11] N. Greene, M. Kass, and G. Miller. Hierarchical Zbuffer visibility. In *SIGGRAPH 1993 Conference Proceedings*, Annual Conference Series, August 1993, pp. 231–238.

[12] W. Heidrich. High-quality shading and lighting for hardware-accelerated rendering. *PhD thesis*, University of Elangen, Germany, 1999.

[13] M. Kilgard. Shadow Mapping with Today's OpenGL Hardware. In *Game Developer's Conference 2000*, San Jose, CA, 2000.

[14] T. Lokovic and E. Veach. Deep Shadow Maps. *SIGGRAPH 2000 Conference Proceedings*, pp. 385-392. Addison Wesley, July 2000.

[15] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. In *SIGGRAPH 87 proceedings*, pages 283-291, 1987.

[16] M. Segal, C. Korobkin R. Van Widenfelt, J. Foran, and P. E. Haeberli. Fast shadows and lighting effects using texture mapping. In *SIGGRAPH 92 proceedings*, pages 249-252, 1992.

[17] S. Upstill. The Renderman Companion: A programmer's guide to realistic computer graphics. *Addison-Wesley, Reading, MA, 1990*.

[18] Y. Wang and Molnar. Second-Depth Shadow Mapping. *UNC-CS Technical Report* TR94-019, 1994.

[19] D. Weiskopf and Ertl. Shadow Mapping Based on Dual Depth Layers. In *Proceedings of Eurographics '03 Short Papers*, pages 53-60, 2003.

[20] T. Whitted. *An improved illumination model for shaded display*. Communications of the ACM, 23(6), pp. 343–349 (1980).

[21] L. Willians. Casting curved shadows on curved surfaces. In *SIGGRAPH 78 proceedings*, pages 270-274, 1978.

[22] L. Willians. Pyramidal Paramerics. In *SIGGRAPH 83 proceedings*, pages 1-1, 1983.

[23] Andrew Woo. The Shadow Depth Map Revisited. In *D. Kirk, editor, Graphics Gems III*, pages 338-342. AP Professional, Boston, 1992.

[24] Andrew Woo, P. Poulin, and A. Fournier. A Survey of Shadow Algorithms. *IEEE Computer Graphics and Applications*: vol 10(6), pages 13-32, 1990.