Eurographics Symposium on Rendering (2007)
Jan Kautz and Sumanta Pattanaik (Editors)

Pixel-Correct Shadow Maps with Temporal Reprojection and Shadow Test Confidence

Daniel Scherzer† Stefan Jeschke Michael Wimmer

Vienna University of Technology, Austria

Abstract

Shadow mapping suffers from spatial aliasing (visible as blocky shadows) as well as temporal aliasing (visible as flickering). Several methods have already been proposed for reducing such artifacts, but so far none is able to provide satisfying results in real time.

This paper extends shadow mapping by reusing information of previously rasterized images, stored efficiently in a so-called history buffer. This buffer is updated in every frame and then used for the shadow calculation. In combination with a special confidence-based method for the history buffer update (based on the current shadow map), temporal and spatial aliasing can be completely removed. The algorithm converges in about 10 to 60 frames and during convergence, shadow borders are sharpened over time. Consequently, in case of real-time frame rates, the temporal shadow adaption is practically imperceptible. The method is simple to implement and is as fast as uniform shadow mapping, incurring only the minor speed hit of the history buffer update. It works together with advanced filtering methods like percentage closer filtering and more advanced shadow mapping techniques like perspective or light space perspective shadow maps.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation - Display algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Virtual reality

1. Introduction

In shadow mapping [Wil78], the shadow computation is performed in two passes: first, a depth image of the current scene (the *shadow map*) as seen from the light source (in *light space*) is rendered and stored. Second, the scene is rendered from the viewpoint (in *view space*) and each 3D fragment is reprojected into the light space. If the reprojected fragment depth is farther away that the depth stored in the shadow map, the fragment is in shadow and shaded accordingly.

Unfortunately, shadow mapping suffers from spatial and temporal aliasing, visible as blocky pixels and flickering, respectively. The primary problem is *undersampling* due to insufficient shadow map resolution. To use the shadow map resolution more efficiently, Brabec et al. [BAS02] focused the shadow map to the visible scene parts, which makes shadow maps useful for larger scenes. The downside is that now the shadow map extents are recalculated each frame and therefore the rasterization is likely to differ each frame, thus creating lots of temporal aliasing (flickering).

The point addressed in this paper is that different subsequent rasterizations are a vast source of information: each rasterization is a discrete approximation of the same (or at least a very similar) continuous depth image that would represent a perfect shadow map. Note that we do not need an infinite resolution shadow map for a fixed resolution view port. Instead, a depth sample in the shadow map at each reprojected view space sample position is sufficient. In practice this results in an irregular grid shadow map [AL04], which can hardly be implemented in current hardware.

In this paper, instead of adapting the shadow map resolution, we accumulate the required information per pixel over time by using a so-called *history buffer* in screen space. This buffer accumulates shadow map test results for the past few frames, reprojected to the current frame. A special confidence-based history buffer update based on the current shadow map ensures that it converges to an exact shad-

† {scherzer|jeschke|wimmer}@cg.tuwien.ac.at

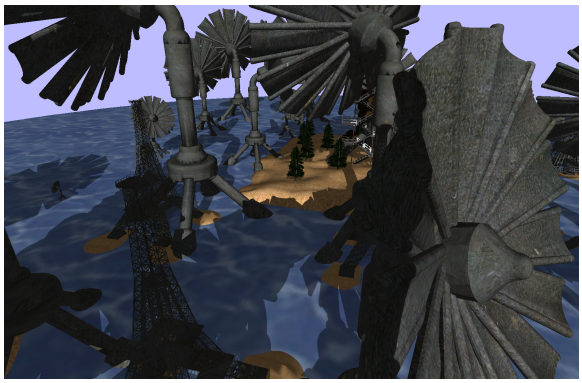

Figure 1: *Pixel-correct shadow maps as a result of using a shadow map (size* 1024^2*) with shadow test confidence together with a history buffer. Note that projection and perspective aliasing are completely removed.*

owing solution. Consequently, shadow borders sharpen over time until both perspective *and* projective aliasing artifacts are completely removed (see Figure 1). Likewise, temporal aliasing (shadow flickering) is removed, since shadows are smoothed during the convergence process. Note that in practice the convergence is typically faster than the eye adaption so that the image quality appears consistently high.

The main contribution of this paper is a shadow mapping algorithm that quickly adapts to pixel correct (hard) shadows over time. Similarly, it eliminates image flickering by smoothing shadows in case of insufficient information. No scene analysis is required, so that the rendering speed is practically not reduced. In addition, the algorithm is simple to implement and integrate into existing rendering engines. It works together with advanced filtering methods like percentage closer filtering and more advanced shadow mapping techniques like perspective or light space perspective shadow maps.

2. Previous Work

Removing the inherent drawbacks of shadow maps is an active research field and therefore a number of papers exists. We divide the approaches into *filtering-based*, *reparameterization-based* and *pixel exact* solutions and name the most relevant representatives.

Filtering: Hard jagged shadow map edges can be substituted with high-frequency noise. For example, percentage closer filtering (PCF) [RSC87] uses Poisson disk sampling in light space to approximate the reprojected eye fragments. An alternative is to use (bi)linear or cubic filtering of numerous shadow map test results to blur shadow map edges. This produces smooth shadow map boundaries which are incorrect but convincing.

Reparameterizations: There exist a number of reparameterization approaches which try to allocate more samples to close regions. *Perspective shadow maps* [SD02] use the post perspective view space to warp the shadow map. *Light space perspective shadow maps* [WSP04] warp the light space with a view-aligned transformation. The approaches have in common that they target perspective aliasing. Perspective aliasing is caused by the mismatch between uniform shadow map resolution and the non-uniform resolution that is caused by perspective eye views.

Pixel exact solutions: *Adaptive shadow maps* [FFBG01] store the shadow map in a hierarchical grid structure that is updated each frame. *Alias-free shadow maps* [AL04] transform all visible eye fragments into light space and provide a hierarchical software implementation to efficiently evaluate shadow information at these (undiscretized) points. *Tiled shadow maps* [Arv04] tile the light view to change the sampling quality according to a heuristical analysis. All these approaches only achieve interactive frame rates, while our method runs at practically the full speed of standard shadow maps.

Especially in off-line ray casting approaches, a number of papers exists that make use of information stored over time [HBS03]. A recent Siggraph sketch [NSI06] argues that this can also be done in real time in a so-called reprojection cache, an idea that seems similar to our history buffer.

3. Pixel-Correct Shadow Maps

The main idea in this paper is to reuse shadowing information of previously rendered frames to increase the shadow accuracy of the current frame while taking into account the *confidence* of this information. We accumulate this information in a so-called *history buffer*, which will be introduced in detail in Section 3.1. The actual shadow mapping operation is then performed in every frame by the following steps:

1. Calculate the shadowing for all fragments of the current frame using standard shadow mapping.
2. Transform the history buffer to the current frame (see Section 3.2). Note that in practice we will do the reverse and transform each fragment from the current frame back into the history buffer.
3. Update the history buffer using the current shadow map (see Section 3.3).
4. Shadow the current frame according to the updated history buffer.

3.1. Temporal Smoothing with the History Buffer

In order to reduce temporal aliasing, we interpret each pixel as a separate function with the time as the input domain (usually represented by a frame number). Temporal anti-aliasing is then done by smoothing this function. An obvious way

would be to use floating averages

$$s(n) = \frac{f(n) + f(n-1) + \ldots f(n-k)}{k} \qquad (1)$$

were k is the number of frames we consider and f is the function we want to smooth. $f(n)$ is the function result for the current frame, $f(n-1)$ for the last frame and so forth. This straightforward approach raises two problems. First, we would need to store $f(x)$ for every pixel for the last k frames, which results in high storage costs. We would prefer to calculate our results incrementally, so that we would only have to store the functions of a single frame. Second, if all frames are weighted equally, arriving at the actual value of the function for the current frame would need k consecutive identical frames, which introduces a latency of k frames. Instead, we want to have increasing weights for more recent frames. This would let the function converge quickly to the current frame while maintaining some smoothing.

Fortunately, the commonly used *exponential smoothing* method circumvents these two drawbacks and is quite fast to compute. It works iteratively and allows adjustable weights:

$$s(n) = w * f(n) + (1-w) * s(n-1) \qquad 0 < w \leq 1 \qquad (2)$$

Here w is a weight and $s(n-1)$ is the result of the previous evaluation. w allows balancing fast adaption of s to changing input parameters against temporal noise of the function. With increasing w, $s(n)$ depends more on the result of the current frame function and less on older frames and vice versa. The name exponential smoothing comes from the exponential falloff of the influence of older frames implied by the recursion. $s(n)$ needs to be defined for each screen pixel

$$s_{x,y}(n) = w * f_{x,y}(n) + (1-w) * s_{x,y}(n-1), \qquad (3)$$

thus defining the so-called *history buffer*. This buffer is simply a screen size array that allows us to capture shadowing information of a potentially infinite number of old frames with exponential falloff. Note that in practice we need to double-buffer this array, since current hardware cannot read from and write to the same buffer in the same frame. The function $f_{x,y}(n)$ is simply the *result* of the shadow map test for each fragment. This test returns 0 for a shadowed fragment and 1 if the fragment is lit. The buffer has a fairly small memory footprint, containing for each fragment $f_{x,y}(n)$ and the depth. The latter is needed for a correct depth buffer lookup (Section 3.2). Also note that the weights w can be calculated individually for each pixel ($w_{x,y}$). This will become important in Section 3.3, where $w_{x,y}$ will be based on the individual pixel history.

3.2. History Buffer Transformation

In practice, when evaluating Equation 3, we have to get the shadowing information from previous frames from the history buffer, namely $s_{x,y}(n-1)$. The history buffer represents shadowing information for 3D fragments in world space. Consequently, if the camera moves, for every currently rendered fragment we have to find the corresponding position in the history buffer (i.e., the previous frame). Since we have the 3D position of our current fragment, we can simply use the view (**V**) and projection (**P**) matrices and their inverses of the current and the last frame to do the transformation:

$$\mathbf{p}_{n-1_{x',y',z'}} = \mathbf{P}_{n-1} * \mathbf{V}_{n-1} * \mathbf{V}_n^{-1} * \mathbf{P}_n^{-1} * \mathbf{p}_{n_{x,y,z}} \qquad (4)$$

The obtained position will normally not be at an exact fragment center in the history buffer except for the special case that no movement has occurred. Consequently, filtering the history buffer for the lookup should be done. In practice, the bilinear filtering that graphics hardware offers shows good results.

An important issue is the treatment of fragments that have no corresponding entry in the history buffer. These are fragments that project to a position outside the history buffer. For instance, such new fragments occur on the left screen border in Figure 2 after a translation to the left. Similarly,

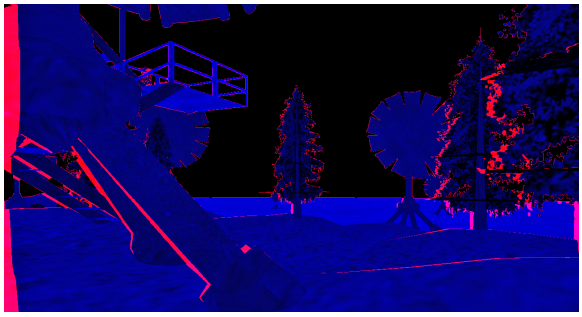

Figure 2: *New fragments (shown in red) without history that result from a camera translation.*

fragments might be missed in the history buffer due to disocclusions if new scene parts appear behind occluders (as for the trees in Figure 2). In order to detect such cases, we check the depth difference between the current fragment and the corresponding history buffer entry. If this distance exceeds a certain threshold, we conclude that this fragment is new and therefore has no history. For all new fragments the original unsmoothed shadowing function is used $s_{x,y}(n) = f_{x,y}(n)$.

3.3. Confidence-Based History Buffer Updates

While temporal smoothing with the history buffer reduces temporal aliasing, shadow edges are only smoothed without improving their accuracy. To achieve this, we take the *confidence* of a sample into account, which can be motivated as follows. The use of a simple shadow map test as source function $f_{x,y}(n)$ for the history buffer shows the following property: If an eye fragment transformed into light space is *exactly* at the center of a shadow map pixel, the corresponding shadow map test result is the correct shadowing solution for that eye fragment. We use this property to associate

with each shadow map test result a *confidence*, which is a measure for the correctness of the test in dependence on the max-norm distance of the transformed eye space fragment to the nearest shadow map sample:

$$conf_{x,y} = 1 - max\Big(|x - center_x|, |y - center_y| \Big) * 2, \quad (5)$$

were $conf_{x,y}$ is the confidence at the fragment position (x,y) and $(center_x, center_y)$ is the pixel center. If we assume a sample size of 1, we get a highest max norm distance 0.5, which explains the factor 2.

The central idea of this paper is now to use this shadow map confidence as input for the weight w for the history buffer update (Equation 3). In combination with the constantly updated history buffer, this allows us to compute pixel exact shadows even with very low resolution shadow maps, provided we have *different rasterizations* of the shadow map in each frame. The reason this works is that each new shadow map sample influences the current shadowing result $s(n)$ only if it has a high confidence. As long as different rasterizations are produced, it is very likely that a "good" shadow test result appears, and this will have a high influence on the shadow result, whereas "bad" shadow test results only have little influence.

Different rasterizations are provided by sub-pixel jittering the light space projection window in the light view plane based on a pseudo random sequence (Halton). Interestingly, translational jittering alone does not provide the required rasterization randomness. This is probably caused by the fact that translational jittering may or may not lead to a different rasterization of the shadow edges. Consequently, we use an additional plane rotation (with the rotation angle being a member of the Halton sequence), which ensures completely different rasterizations of shadow edges for each frame.

We apply a power function to the confidence

$$w_{x,y} = conf_{x,y}^m, \quad (6)$$

obtaining a single intuitive parameter m to balance fast history buffer adaptation against temporal noise, as mentioned in Section 3.1. If m is chosen relatively low (around 3), the history buffer (and thus shadowing) adaption performs quickly, but some temporal noise remains. If m is chosen relatively high (about 15), the history buffer adapts slowly, but shows no temporal noise. In this case, it is strongly biased towards more accurate shadow map samples, converging to practically pixel exact shadows. The temporal noise is visible as unsharp shadow edges that vary over time. In practice we use a low value of m if the camera is moving and start increasing m as soon as it stands still. This lets the shadow map adapt similarly to the eye, making the convergence practically invisible if the framerate is above 30Hz (see also Figure 3 and the video provided with this paper).

4. Implementation and Results

With vertex and pixel shader support, the method is efficient and simple to implement: the reprojection matrix is multiplied with the post-perspective vertex position in the vertex shader and interpolated. The only thing left to do in the pixel shader is to use this position for the lookup in the history buffer.

All images were taken with a shadow map resolution of 1024^2. Frame buffer objects were used to render the history buffer in 16bit floating point format. Depth was stored with the same precision. Two instances of the history and depth buffer are required (one for reading and one for writing), resulting in an overall memory requirement of only 4MB for a screen resolution of 1024^2. Frame buffer and history buffer are written in one pass using the multiple render target functionality. Minor speed penalties of the method occur primarily in the fragment shader, where a read and a write to the history buffer is necessary. On a Pentium4 3.2GHz with NVIDIA GeForce 8800GTX graphics card, for a 1024^2 window we measured an average frametime of 15ms for the scene shown in Figure 4, of which about 1.5ms is the overhead incurred by our method.

All images shown of our method use a value of $m = 15$ for the weight calculation. The accompanying video uses $m = 3$ while moving and increases m each frame of non-movement by 0.1. Figure 3 shows the convergence properties of our method when standing still for varying strategies of choosing m for example viewpoints. The y-axis shows the total pixel error as the percentage of pixels differing from the fully converged solution, counting only pixels with a maximum difference greater than 30 in RGB (with a range from 0–255). The x-axis is in frames. At frame 0 we already have an error of only 0.25% with preceding coherent camera movement, quickly converging to 0.1% error in 20 frames. Typically this error is not noticeable anymore. Figures 4 and 6 show comparisons of light space perspective shadow maps and our method (see also Figure 1 on the colorplate). Note especially in Figure 4 the dueling frusta case. As Figure 6 shows, using PCF-filtered lookup results for $f_{x,y}(n)$ creates smoother shadow borders. If we use a largely undersampled shadow map representation, as for example a uniform shadow map in a large scene, convergence to the pixel exact solution is slow. We greatly increased the convergence by using light space perspective shadow maps, which already reduce perspective aliasing errors. With this we can provide the pixel exact solution in a few iterations (about 10 to 60). Figure 5 shows the convergence for an example viewpoint. It is important to note that through the exponential smoothing, we have already removed all flickering artifacts, and the convergence is extremely smooth. In our experiments, the pixel exact solution is quickly reached when an observer stops to investigate shadow map edges in detail (see the enclosed video).

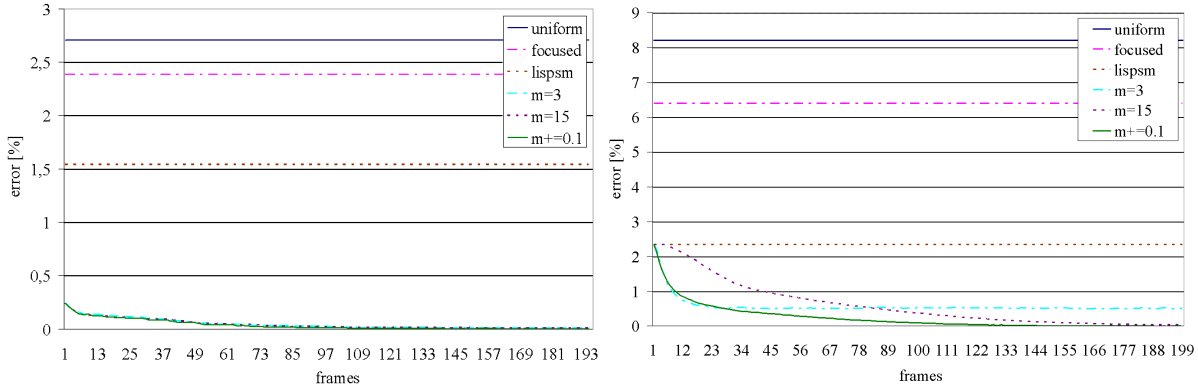

Figure 3: *This figure shows the convergence for two example viewpoints while standing still for different strategies of choosing m. The* left *graph shows the behavior after a preceding coherent rotation of the camera (filled history buffer) and the* right *graph shows the behavior when starting with an empty history buffer.*

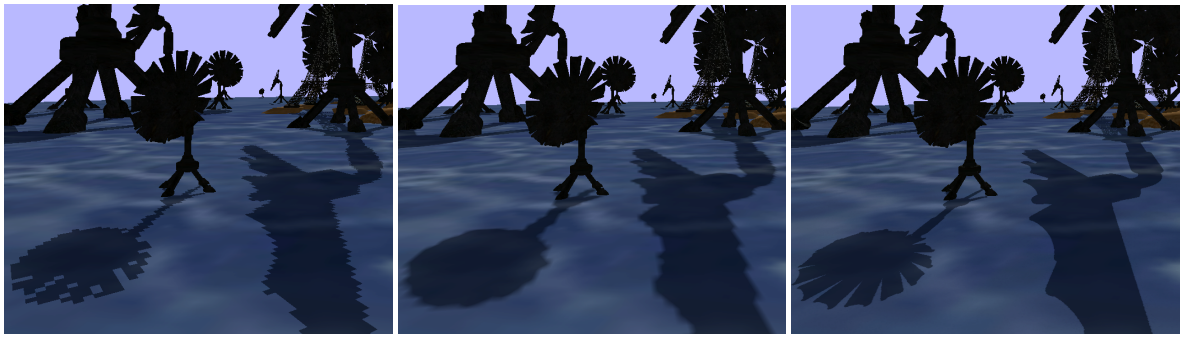

Figure 4: *The famous dueling frusta case where reparameterizations of the shadow map can only provide similar results as uniform shadow maps. On the left: light space perspective shadow maps; in the middle: light space perspective shadow maps with PCF 3x3 filtering; and on the right: our new method.*

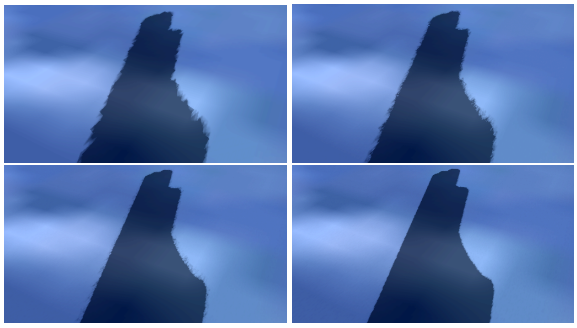

Figure 5: *Shadow adaption over time after 1 (top-left), 20 (top-right), 40 (bottom-left)and 60(bottom-right) frames.*

5. Conclusions and Future Work

In this paper we presented a new shadowing method based on shadow maps that produces pixel accurate shadows in real time. It resolves perspective and projective aliasing, which are present in all existing real-time shadow algorithms. Furthermore this shadow method creates temporally smooth shadow transitions, thus resolving temporal aliasing and flickering artifacts that are a major problem of shadow mapping. In addition, the algorithm is simple to implement and to integrate into existing rendering engines. It works together with advanced filtering methods like percentage closer filtering and more advanced shadow mapping techniques like perspective or light space perspective shadow maps, and has practically no performance overhead over standard shadow mapping.

A possible avenue for future work is to extend the idea to soft shadows. Here the jittering would have to be applied to positions on the area light source to create physically correct soft shadows. This approach would include the real 3D visibility solution for area/volume light sources, which has not been solved so far in real time. Another important issue we want to study are dynamic scenes. Our optimizations of *m*

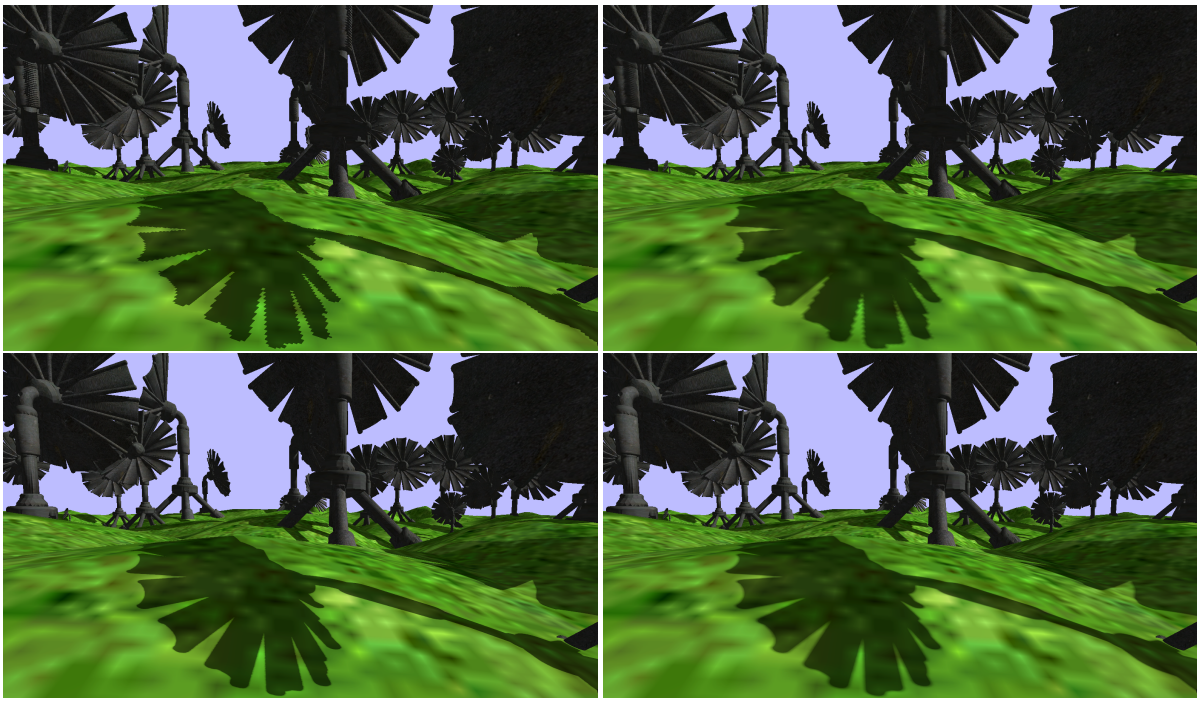

Figure 6: *Light space perspective shadow maps (top-left). Our new pixel correct shadow method (bottom-left). To the right both methods with 3x3 PCF filtering to produce fake soft shadow borders. LispSM (top-right) and our pixel correct shadow method below. Please note that not even PCF filtering can hide the projection aliasing artifacts and false inter-object shadows present in the scene shadowed with LispSM.*

distinguish only between a moving or a still observer. This model would have to be adapted to include moving objects and light sources.

References

[AL04] AILA T., LAINE S.: Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004* (2004), Eurographics Association, pp. 161–166.

[Arv04] ARVO J.: Tiled shadow maps. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 240–247.

[BAS02] BRABEC S., ANNEN T., SEIDEL H.-P.: Practical shadow mapping. *Journal of Graphics Tools: JGT 7*, 4 (2002), 9–18.

[FFBG01] FERNANDO R., FERNANDEZ S., BALA K., GREENBERG D. P.: Adaptive shadow maps. In *SIGGRAPH 2001 Conference Proceedings* (Aug. 2001), Fiume E., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 387–390.

[HBS03] HAVRAN V., BITTNER J., SEIDEL H.-P.: Exploiting temporal coherence in ray casted walkthroughs. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics* (New York, NY, USA, 2003), ACM Press, pp. 149–155.

[NSI06] NEHAB D., SANDER P. V., ISIDORO J. R.: The real-time reprojection cache. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches* (New York, NY, USA, 2006), ACM Press, p. 185.

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. Stone M. C., (Ed.), vol. 21, pp. 283–291.

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Siggraph 2002 Conference Proceedings* (July 2002), vol. 21, 3, pp. 557–562.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings) 12*, 3 (Aug. 1978), 270–274.

[WSP04] WIMMER M., SCHERZER D., PURGATHOFER W.: Light space perspective shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004* (2004).