Volume 0 (1981), Number 0 pp. 1–17 COMPUTER GRAPHICS *forum*

A Survey of Real-Time Hard Shadow Mapping Methods

Daniel Scherzer, Michael Wimmer and Werner Purgathofer†

Vienna University of Technology

Abstract

Due to its versatility, speed and robustness, shadow mapping has always been a popular algorithm for fast hard shadow generation since its introduction in 1978, first for off-line film productions and later increasingly so in real-time graphics. So it is not surprising that recent years have seen an explosion in the number of shadow map related publications. The last survey that encompassed shadow mapping approaches, but was mainly focused on soft shadow generation, dates back to 2003 [HLHS03], while the last survey for general shadow generation dates back to 1990 [WPF90]. No survey that describes all the advances made in hard shadow map generation in recent years exists.

On the other hand, shadow mapping is widely used in the game industry, in production, and in many other applications, and it is the basis of many soft shadow algorithms. Due to the abundance of articles on the topic, it has become very hard for practitioners and researchers to select a suitable shadow algorithm, and therefore many applications miss out on the latest high-quality shadow generation approaches.

The goal of this survey is to rectify this situation by providing a detailed overview of this field. We provide a detailed analysis of shadow mapping errors and derive a comprehensive classification of the existing methods. We discuss the most influential algorithms, consider their benefits and shortcomings and thereby provide the readers with the means to choose the shadow algorithm best suited to their needs.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Picture/Image Generation—Display algorithms; Computer Graphics [I.3.3]: Picture/Image Generation—Viewing algorithms; Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism - Virtual reality—Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism - Color, shading, shadowing, and texture—

1. Introduction

Shadows are an important result of the light transport in a scene. They give visual cues for clarifying the geometric relationship between objects and between objects and light sources. While soft shadows due to area light sources are becoming increasingly popular in applications like games, many applications still use hard shadows, which are caused by a point light or a directional light. Even if soft shadows are used for some light sources in an application, many light sources can be modeled acceptably well as point lights giving hard shadows or shadows that are slightly softened using filtering techniques (an example for this is the shadow caused by the sun). Our survey will focus on hard shadows, because they are the most widely used shadow algorithm, but their potential is rarely fully exploited because of the abundance of papers on the subject, which makes it difficult to choose the best algorithm for a particular application.

A point is in shadow when this point cannot be seen from the viewpoint of the light source. The object which blocks the light rays from reaching this point is called the shadow caster, occluder or blocker. The object on which the point in shadow lies is called the shadow receiver (see Figure 1). Two major approaches to real-time hard shadows exist: *geometry-based* and *image-based*.

Even though shadow algorithms have been around for almost as long as computer graphics itself, robust and efficient hard shadow generation is still not a solved problem. While geometry-based algorithms produce pixel-perfect results, they suffer from robustness problems with different viewer-light constellations. Due to their versatility, al-

† e-mail: scherzer | wimmer | wp@cg.tuwien.ac.at

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

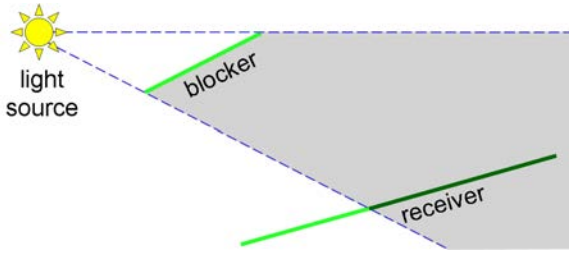

Figure 1: *The geometry of shadow casting.*

most all research in geometry-based algorithms focuses on *shadow volumes* [Cro77]. The main disadvantage of this technique is the vast amount of fill-rate needed to render all the shadow volumes. Additionally a silhouette detection has to be made, for polygon-rich scenes this means another performance penalty. Finally, only polygonal data can be processed, because a simple way to detect and extrude edges is needed.

Image-based algorithms, on the other hand, are very fast as their complexity is similar to standard scene rendering. *Shadow mapping* [Wil78] is an image-based algorithm that can handle arbitrary caster/receiver constellations, can account for self shadowing and can even process non polygonal input. The basic shadow algorithm is covered in Section 2.

Unfortunately, shadow mapping also suffers from a number of drawbacks. First, omni-directional lights cannot be captured using a single frustum. This issue is discussed in Section 2.

A second and more severe problem are aliasing artifacts that arise because the sampling of the shadow map and the sampling of the image pixels projected into the shadow map usually do not match up. In Section 3 we will analyze these aliasing artifacts and derive a number of different types of error from this analysis, while Section 4 gives an overview of methods that can reduce the sampling error.

A third problem, incorrect self-shadowing, is caused by undersampling and imprecisions in the depth information stored in the shadow map for each texel. This creates the need to bias the depth test (*depth biasing*) to give robust results. We will discuss various approaches to this problem in Section 5.

Section 6 introduces filtering techniques that apply sampling theory to better reconstruct the information stored in the shadow map in the second pass. Finally, Section 7 gives guidelines on how to choose the best algorithm for a given application.

2. Basics

In shadow mapping 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 (see Figure 2, left). This image contains for each texel the depth of the nearest object to the light source. The idea is that everything that lies behind those depths cannot be seen by the light source and is therefore in shadow. In the second pass, 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 than the depth stored in the shadow map (*depth test*), the fragment is in shadow and shaded accordingly (see Figure 2, right).

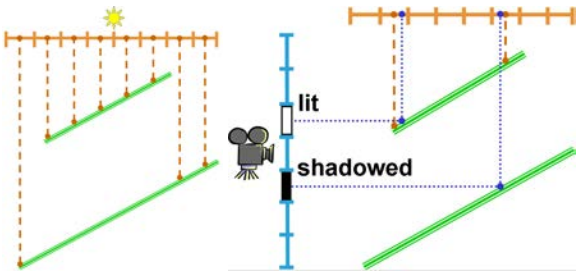

Figure 2: *Shadow mapping: First, a depth image (the* shadow map*) of the scene is generated by rendering from the viewpoint of the light source* (left)*. Second, the reprojected depth of each view space fragment is compared to the depth stored in the shadow map* (right)*.*

Omni-directional lights have to be calculated by using multiple buffers due to their spherical view. No single frustum can reflect this, and so a number of shadow maps and frusta have to be built to divide this spherical view. The most common approach uses six frusta (one for each side of a cubemap), which causes a big performance penalty for such lights. A faster solution can be achieved by employing a parabolic mapping [BAS02b]. This results in only two renderings, one for each hemisphere, but also creates the problem of how to mimic the parabolic mapping (lines become curves) efficiently on graphics hardware. The simplest solution is to assume that the scene is tessellated finely enough so that a parabolic mapping of the vertices alone is sufficient. Slower and more involved approaches exist that calculate the curves directly on modern hardware [GHFP08].

3. Error Analysis

When rendering a shadow map, a discretely sampled representation of a given scene is created. The shadow mapping operation later uses this representation to reconstruct the visibility conditions of surfaces with respect to the light source. Therefore it is helpful to think about shadow mapping as a signal reconstruction process similar to texture mapping. Signal reconstruction has the following steps:

1. Initially *sample* an input function, i.e., generate the shadow map using rendering. Since no bandlimiting is possible to avoid aliasing in the initial sampling phase, the sampling frequency should ideally be higher than the Nyquist frequency of the signal. Note that the actual shadow signal after evaluating the depth comparison has sharp edges and therefore unlimited frequencies.
2. *Reconstruct* (or interpolate) the signal from its sampled representation. For shadow mapping, this happens when projecting a fragment into shadow map space and evaluating the shadow map test. Due to the non-linearity of the depth test, only a nearest-neighbor lookup is possible for straightforward shadowmapping.
3. *Bandlimit* (or prefilter) the reconstructed signal to avoid too high frequencies at the targeted output resolution. This is not straightforwardly possible for shadow mapping due to the non-linearity of the depth test.
4. *Resample* the reconstructed signal at the final pixel positions.

Note that in discrete sampling scenarios, reconstruction happens only at the final pixel positions, so reconstruction and resampling are in a way combined.

In order to facilitate a mental model of shadow mapping as signal reconstruction process with the above steps, one can think about the shadow mapping operation as texturing objects with a projective texture map that has the result of the shadow test for that particular object already baked in. This means that the texels are filled with 1 for fragments in light and 0 for fragments in shadow. In this way, reconstruction and resampling can be compared to standard shadow mapping, where reconstruction is done with a nearest-neighbor lookup or bilinear interpolation, bandlimiting is done using mipmapping, and resampling is just the evaluation of the function at the final pixel positions.

3.1. Types of Error

The main types of error are

- Undersampling, which occurs when the shadow map samples projected to the screen have a lower sampling rate than the screen pixels. This is due to a too low initial sampling frequency.
- Oversampling, which happens when the shadow map samples projected to the screen have a higher sampling rate than the screen pixels. In this case, the classical aliasing known from texture sampling occurs. This can be fixed either by adapting the initial sampling rate, or by the bandlimiting/prefiltering step discussed above (see Section 6).
- Reconstruction error or staircase artifacts, which are due to nearest neighbor reconstruction. This can be fixed by better reconstruction filters (see Section 6).
- Temporal aliasing or flickering artifacts, if the rasterization of the shadow map changes each frame. These artifacts will appear especially for non-optimal reconstruction if undersampling occurs, see Section 4.6 for a remedy.

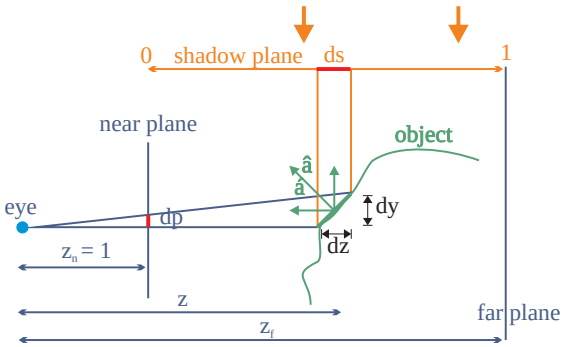

Figure 3: *Aliasing in shadow mapping.*

The most important difference to signal processing or texture mapping in particular is that the process of shadow mapping gives some control over the initial sampling step. Changing the initial sampling reduces the effects of all types of error, and thus most publications on shadow mapping take this approach. In Section 3.2 and 3.3, we will therefore discuss *sampling error* in more detail and discuss two methods to characterize this error type. There are also some approaches targeted specifically at dealing with oversampling and reconstruction in the context of shadow mapping, which we will describe in Section 6.

3.2. Simplified Sampling Error Analysis

At the root of most algorithms to reduce shadow map sampling errors is an analysis of the distribution of errors in a scene. A simplified error analysis was first introduced by Stamminger and Drettakis [SD02] for Perspective Shadow Maps, and the same formula is used in many subsequent approaches. The analysis assumes an overhead directional light and looks at a surface element located somewhere on the z-axis of the view frustum. Figure 3 shows a configuration for a small edge.

A pixel in the shadow map represents a shaft of light rays passing through it and has the size $ds \times ds$ in the local parametrization of the shadow map. We assume a local parametrization of the shadow map which goes from 0 to 1 between near and far planes of the viewer – this already assumes that the shadow map is properly focused to the view frustum, not wasting any resolution on invisible parts of the scene (see Section 4). In world space, the shaft of rays has the length $dz = (z_f - z_n)ds$ for uniform shadow maps as an example.

The shaft hits a small edge along a length of $dz/\cos\beta$.

This represents a length of $dy = dz\frac{\cos\alpha}{\cos\beta}$ in eye space, projecting to $dp = dy/z$ on screen (assuming a near plane distance of 1). Note that we assume that the small edge can be translated along the z-axis, i.e., z is the free parameter of the analysis. The shadow map aliasing error dp/ds is then

$$\frac{dp}{ds} = \frac{1}{z}\frac{dz}{ds}\frac{\cos\alpha}{\cos\beta}. \tag{1}$$

Shadow map *undersampling* occurs when dp is greater than the size of a pixel, or, for a viewport on the near plane of height 1, when dp/ds is greater than $res_{shadowmap}/res_{screen}$. As already shown by Stamminger and Drettakis [SD02], this can happen for two reasons: *perspective aliasing* when $\frac{dz}{zds}$ is large, and *projection aliasing* when $\cos\alpha/\cos\beta$ is large.

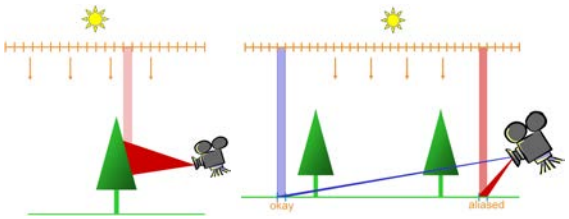

Figure 4: *In the figure on the* left *side the cause for projection aliasing is the orientation of the tree's surface: It projects to a small area in the shadow map, but projects to a big area in camera space. Perspective aliasing on the* right side *occurs because the shadow map is indifferent to perspective foreshortening and distant as well as near areas (with respect to the camera) are therefore stored with the same resolution, but project to very different sizes in camera space.*

Projection aliasing is a local phenomenon that occurs for surfaces almost parallel to the light direction (see Figure 4, left). Reducing this kind of error requires higher sampling densities in such areas. Only approaches which adapt the sampling density locally based on a scene analysis can achieve this (Sections 4.3.3 to 4.6).

Perspective aliasing, on the other hand, is caused by the perspective projection of the viewer (see Figure 4, right). If the perspective foreshortening effect occurs along one of the axes of the shadow map, it can be influenced by the *parametrization of the shadow map*. If a different parametrization is chosen, this will lead to a different sampling density distribution along the shadow map. The standard uniform parametrization has dz/ds constant, and therefore the sampling error dp/ds is large when $1/z$ is large, which happens close to the near plane. This results in a small number of samples spent on objects near the camera, which makes the effects of this error very noticeable. (compare Figure 5). In order to reduce perspective aliasing, there are several approaches to distribute more shadow map samples near the viewer, either by using a different parametrization, or by splitting the shadow map into smaller parts (Sections 4.2 and 4.3).

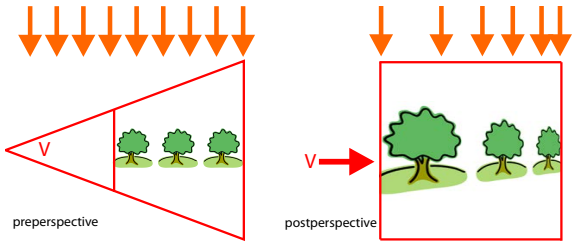

Figure 5: *The uniform distribution of a shadow map in world space (left) degrades near the observer due to perspective foreshortening. This effect is visible in post-perspective space (right). Much fewer samples are spent on nearby elements.*

3.3. Accurate Sampling Error Analysis

An accurate analysis of sampling error is complex and is studied in Brandon Lloyd's article and thesis [Llo07, LGQ*08]. Here we just give the result. For a general configuration, the aliasing error m is

$$m = \frac{r_j}{r_t}\frac{dG}{dt}\frac{W_l}{W_e}\frac{n_e}{n_l}\frac{d_l}{d_e}\frac{\cos\phi_l}{\cos\phi_e}\frac{\cos\psi_e}{\cos\psi_l}. \tag{2}$$

In this formulation (see also Figure 6),

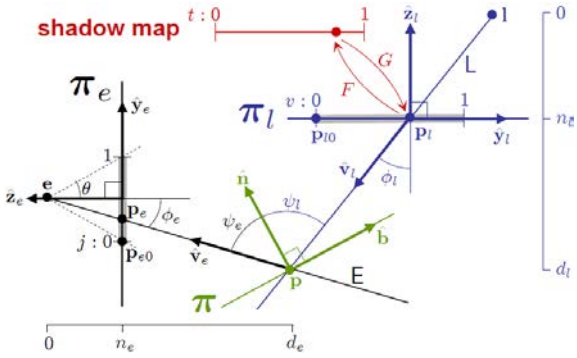

Figure 6: *Notation used in the accurate aliasing description (image courtesy of Brandon Lloyd).*

- $\frac{r_j}{r_t}$ is the ratio of the screen and shadow map resolutions
- $\frac{dG}{dt}$ is the derivative of the shadow map parametrization (called dz/ds above)
- $\frac{W_l}{W_e}$ is the ratio of the world space widths of the light and eye viewports
- $\frac{n_e}{n_l}$ is the ratio of the near plane distances of eye and light
- $\frac{d_l}{d_e}$ is the ratio of the patch distances from the light and from the eye (d_e corresponds to z above)

- ϕ_l, ϕ_e are the angles of the light and eye beams from the image plane/shadow map plane normals
- ψ_l, ψ_e are the angles between light and eye beams from the surface normal of the patch (corresponding to α, β above).

In comparison to the simplified analysis, this formulation takes into account the variations in sampling error when the surface element is not in the center of the view frustum, and for arbitrary light positions or directions. It also correctly accounts for point lights. For directional lights, n_l/d_l converges to 1 and $\cos\phi_l$ will be constant. Shadow map undersampling occurs when $m > 1$.

An important point to consider is that these formulations only treat one shadow map axis. However, reparametrizing the z-axis using a perspective transform also influences the sampling error along the other shadow map axis: if you look at Figure 5, this is the axis orthogonal to the plane this paper is printed on. Texels along this axis also get stretched through reparametrization, affecting sampling error as well. We will shortly consider this effect in Section 4.2.2.

After this analysis, we will now investigate the various solutions to each shadow map error. In the following section we start with the sampling error.

4. Strategies to Reduce the Sampling Error

Unlike texture mapping, where the resolution of the input image is usually predetermined, in shadow mapping there is significant control over the original sampling step. Therefore, it is possible to adapt the sampling so that the projected shadow map samples correspond much better to the screen space sampling rate than naive shadow mapping.

In particular, for a *magnification* scenario, most of the burden lies on the reconstruction filter for texture mapping, whereas for shadow mapping, this burden can be reduced by increasing the sampling rate and thus removing the magnification (or undersampling) from affected areas, so that even nearest neighbor reconstruction can sometimes give good quality. Furthermore, in a *minification* scenario, e.g., for areas which appear small on screen, the initial sampling rate can be reduced, thereby avoiding the need for a costly bandlimiting filter.

Due to the huge amount of literature in sampling error reduction techniques, we further subdivide approaches that try to remove the sampling error into *focusing*, *warping-based*, *partitioning-based* and *irregular sampling-based* algorithms.

4.1. Focusing

One of the most straightforward ways in which the sampling rate can be improved is to make sure that no shadow map space is wasted on invisible scene parts. Especially in outdoor scenes, if a single shadow map is used for the whole scene, then only a small part of the shadow map will actually be relevant for the view frustum. Thus, fitting or focusing techniques, first introduced by Brabec et al. [BAS02a], fit the shadow map frustum to encompass the view frustum.

The geometric solution is to calculate the convex hull of the view frustum and the light position (for directional lights this position is at infinity) and afterwards clip this body with the scene bounding volume and the light frustum (see [WSP04, WS06] for details). Clipping to the scene bounding volume is necessary because today very large view frusta are common and they frequently extend outside the scene borders. We call the resulting body the intersection body B (see Figure 7).

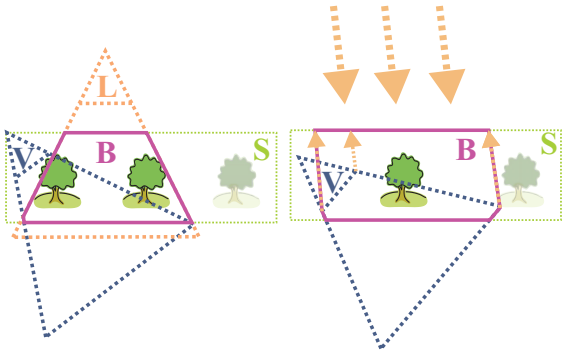

Figure 7: *Shadow map focusing better utilizes the available shadow map resolution by combining light frustum* **L**, *scene bounding box* **S** *and view frustum* **V** *into the bounding volume* **B**. *Here shown on the* left *for point lights and on the* right *for directional light sources.*

The intersection body can be further reduced by using visibility algorithms. If, before the shadow map is created, a first depth-only pass is rendered with an online visibility algorithm like coherent hierarchical culling (CHC) [MBW08], the far plane distance can be reduced to just cover the furthest visible object.

In general, fitting leads to temporal aliasing because the rasterization of the shadow map changes each frame. Especially when using visibility information, strong temporal discontinuities can occur, so using a good reconstruction filter is very important in this case.

Temporal aliasing due to fitting can also be somewhat reduced by trying to keep texel boundaries constant in world space. First, the shadow map needs to maintain a constant orientation in world space in order to avoid projected shadow map texels to change shape whenever the viewer rotates. For this, the shadow map needs to be focused on the axis-aligned bounding box of the intersection body. To avoid aliasing due to translation of the view frustum in the shadow map view, the shadow map should be created with one texel border and

only refit if the view frustum moves a whole texel. However, most viewer movements also lead to a scaling of the view frustum in the shadow map view, and this is more difficult to control without wasting much shadow map space, see [ZZB09] for more details.

4.2. Warping

When projecting the view frustum into the shadow map, it becomes apparent that higher sampling densities are required near the viewpoint and lower sampling densities far from the viewpoint. In some cases, it is possible to apply a single transformation to the scene before projecting it into the shadow map such that the sampling densities are globally changed in a useful way (see Figure 8). In the original algorithms, warping was applied to a single shadow map, however later on it has been combined with partitioning algorithms to further improve sampling rates (see Sections 4.3 and 4.4).

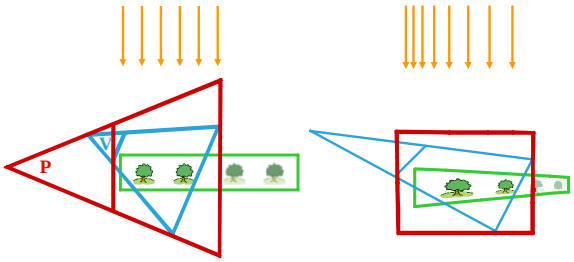

Figure 8: *An example configuration of light space perspective shadow maps with view frustum V and the frustum defining the perspective transform P.* Left: *directional light, a view frustum V, and the perspective transformation P.* Right: *after the warp, objects near the viewer appear bigger in the shadow map and therefore receive more samples.*

Stamminger and Drettakis introduced shadow map warping in their *perspective shadow maps* (PSM) [SD02] paper. The main idea is to apply a perspective transformation, the viewer projection, to the scene before rendering it into the shadow map. Thus, the distribution of shadow map samples is changed so that more samples lie near the center of projection and less samples near the far plane of the projection. This has the benefit that just a simple perspective transformation is used, which can be represented by a 4×4 matrix. This maps well to hardware and is fast to compute. The main problem of this approach is that the achievable quality of this method is strongly dependent on the near-plane of the eye-view, because the error is distributed unevenly over the available depth range. With a close near plane most of the resolution is used up near the eye and insufficient resolution is left for the rest of the shadow map. The authors suggest to analyze the scene to push the near plane back as far as possible to alleviate this problem. Additionally, the use of the viewer projection can change the direction of the light or even the type of the light (from directional to point or vice versa), which complicates implementation.

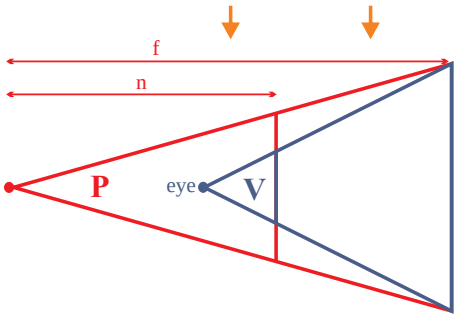

Figure 10: *The parametrization of light space perspective shadow maps (shows the yz-plane in light space). The parameter n is free and can vary between* z_n *(perspective shadow mapping) and infinity (uniform shadow mapping).* **P** *is the perspective transform used for LiSPSM with near plane distance* n *and far plane distance* f. **V** *is the view frustum.*

These problems are circumvented by decoupling the perspective transformation from the viewer. This is the main idea of *light space perspective shadow maps* (LiSPSM) [WSP04], which warp the light space with a light- and view aligned transformation. Here the perspective transformation is always aligned to the axis of the light frustum, and therefore lights do not change direction or type (see Figures 8 and 10). In order to deal with point lights, the projection of the point light is applied first, converting the point light to a directional light, and LiSPSM is done in the post-perspective space of the light. The decoupled perspective transformation has the additional benefit of creating a free parameter, namely the near plane distance n of the perspective transformation (see Figure 9). A small distance leads to a stronger warp and more focus on nearby objects, a larger n leads to a less strong warp. Please note that most of the later work on warping methods has adopted this framework. In LiSPSM the near plane distance is chosen in a way to distribute the error equally over the available depth range, creating homogeneous quality (see Figure 11).

A very similar approach are Martin and Tan's *trapezoidal shadow maps* (TSM) [MT04], which use a heuristic to choose the near plane distance. In a very insightful work, Lloyd et al. [LTYM06] proved that all perspective warping algorithms (PSM, LiSPSM, TSM) actually lead to the same *overall* error when considering both shadow map directions, but LiSPSM gives the most even distribution of error among the directions and is therefore advantageous.

Chong and Gortler [Cho03, CG04, CG07] optimize shadow map quality for small numbers of planes of interest by using multiple shadow maps. They even show that it

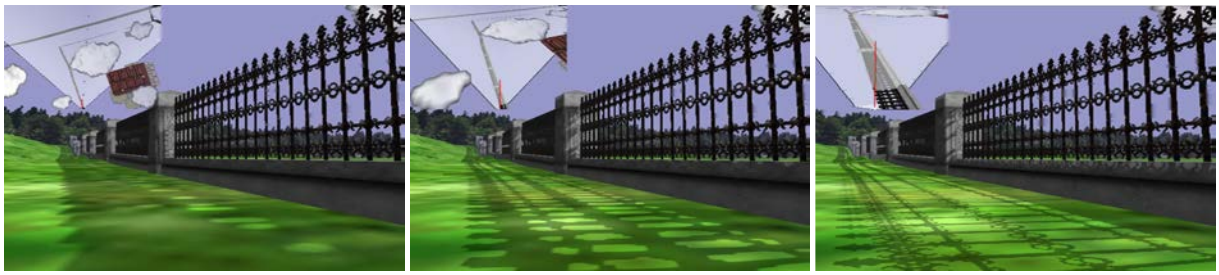

Figure 9: *Decreasing the values* (left *to* right) *of the free parameter n provides increasing resolution of the shadow map near the viewer. Each image shows the warped light view in the upper-left corner.*

is possible to sample those planes perfectly (for each viewport pixel on such a plane, a texel in the shadow map samples exactly the same geometric point) by using a "plane-stabilization" technique from computer vision. Nevertheless, pixels not on such planes can exhibit arbitrary errors.

4.2.1. Logarithmic Warping

Consider again the simplified error formulation shown in Equation 1. An *optimal parametrization* would make dp/ds constant ($= 1$ assuming equal screen and shadow map resolutions) over the whole available depth range. For the ideal case of view direction perpendicular to light direction, this is (constants notwithstanding) equivalent to [WSP04]

$$ds = \frac{dz}{z}, \text{ i.e., } s = \int_0^s ds = \int_{z_n}^{z} \frac{dz}{z} = \ln \frac{z}{z_n}.$$

This shows that the optimal parametrization for shadow mapping (at least for directional lights) is logarithmic. In more recent work, Lloyd at al. [LGQ*08] have revisited the logarithmic mapping and combined it with a perspective warp (LogPSM). In a very involved mathematical treatise, they derive warping functions that approach the optimal constant error very closely, based on the exact sampling error formulation from Equation 2. They also consider fully general 3D configurations.

Unfortunately, such a parametrization is not practical for implementation on current hardware: The logarithm could be applied in a vertex program, however, pixel positions and all input parameters for pixel programs are interpolated hyperbolically. This makes graphics hardware amenable to perspective mappings, but not logarithmic ones. As a proof of concept, logarithmic rasterization can be evaluated exactly in the fragment shader by rendering quads that are guaranteed to bound the final primitive, but this is too slow for practical implementation. To alleviate this, Lloyd et al. [LGMM07] propose simple modifications to the rasterization pipeline to make logarithmic rasterization feasible, but this modification is not likely to be implemented in graphics hardware until rasterization itself becomes a programmable component in the future.

4.2.2. Optimal Warping Parameter for Perspective Warping

As mentioned above, there is a free parameter for P in perspective warping methods, namely, the distance n of the projection reference point $\vec{p}$ to the near plane. This parameter influences how strongly the shadow map will be warped. If it is chosen close to the near plane of P, perspective distortion will be strong, and the effect will resemble the original perspective shadow maps (where n is chosen the same as the view frustum near plane distance). If it is chosen far away from the far plane of P, the perspective effect will be very light, approaching uniform shadow maps. It can be shown that in the case of a view direction perpendicular to the light vector, the optimal choice for this parameter is [WSP04]

$$n_{\text{opt}} = z_n + \sqrt{z_f z_n},$$

where z_n and z_f are the near and far plane distances of the eye view frustum. Figure 12 compares the aliasing error along the viewer z-axis for uniform shadow maps, perspective shadow maps with a warping parameter as in the original PSM paper, and the optimal warping parameter. Note, however, that this analysis only treats errors in the shadow map direction aligned with the z-direction. Considering the x-direction, the PSM parameter actually leads to an optimal constant error, however, as can be seen in the plot, the error along the z-direction is very uneven and leads to very bad shadow quality when moving away from the viewer. The optimal LiSPSM parameter leads to an even distribution of errors among the two axes [LGQ*08].

When the viewer is tilted towards the light or away from it, n has to be increased, so that it reaches infinity when the viewer looks exactly into the light or away from it. In this case, perspective warping cannot bring any improvements, therefore no warping should be applied.

A more involved falloff function that creates more consistent shadow quality has been proposed in [ZXTS06]. Still, it only takes errors along the z-axis into account and was therefore later superseded by Lloyd's [Llo07] approach.

In the original LiSPSM paper, a falloff depending on the angle γ between the shadow map normal vector and the view

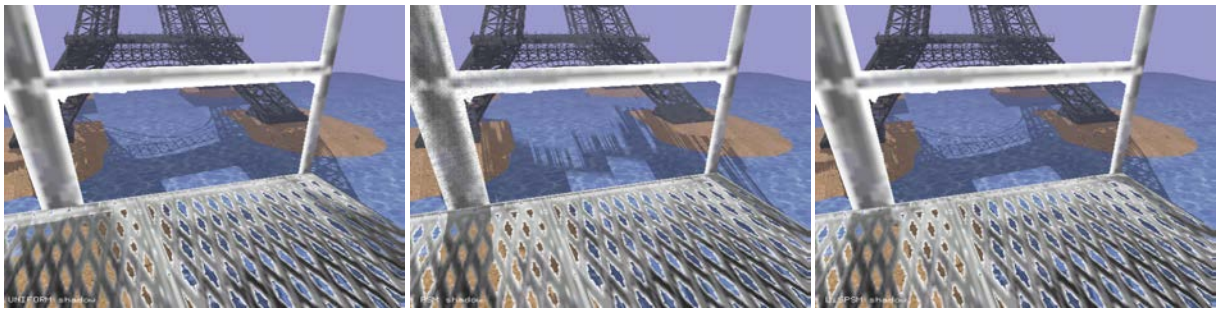

Figure 11: *Comparison of uniform* (left)*, perspective* (middle) *and light space perspective shadow maps* (right)*, each using a* 1024^2 *shadow map.*

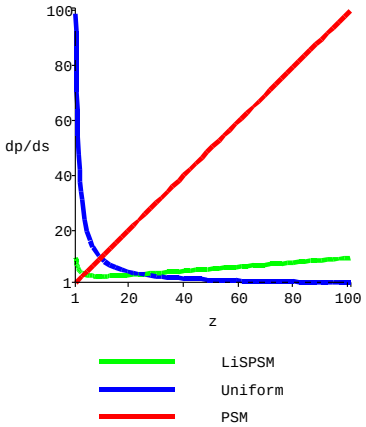

Figure 12: *Perspective aliasing errors plotted against z-coordinate for different shadow mapping techniques for an overhead directional light.*

plane normal vector was introduced, by $n'_{\text{opt}} = n_{\text{opt}} / \sin\gamma$. However, Lloyd [Llo07] later showed that this falloff is not fast enough once the angle passes the point where one of the view frustum planes becomes parallel to the shadow map normal. He proposes a different falloff function that avoids this problem. It is a bit too involved to reproduce here, so we refer the reader to Section 5.1.2.1 and 5.2.1 of [Llo07] for the exact equations. The formulation of Lloyd is therefore more robust in general cases that can occur in practical scenarios.

Another interesting extension is to use a different view frustum near plane distance for the computation of n. The rationale is that the nearest depth values (e.g., between 0.01 and 1) often do not contain visible shadows, but a lot of precision in the shadow map is wasted on this range using the optimal warping parameter. Lloyd describes using a pseudo-near plane in his thesis in Section 5.1.9.

Still, despite these improvements, the main problem of warping-based approaches remains: When the angle γ decreases and approaches 0 (view and light directions become parallel), warping becomes ineffective. In this case, one global perspective warp cannot change the sampling densities along the z-axis of the viewer, and therefore warping degenerates to uniform shadow mapping. This makes this class of approaches susceptible to rapid shadow quality changes, which can produce temporal flickering artifacts. The partitioning methods described in the next section are more robust in this respect.

4.3. Partitioning

In contrast to warping methods, partitioning methods try to approximate the ideal sample distribution by using multiple shadow maps.

4.3.1. Z-Partitioning

The most prominent approach and one of the most practical algorithms is to subdivide the view frustum along the z-axis, and calculate a separate equal-sized shadow map for each sub-frustum. This algorithm goes by the names of *plural sunlight buffers* [TQJN99], *parallel split shadow maps* (PSSM) [ZSXL06], *z-partitioning* [LTYM06] or *cascaded shadow maps* (CSM) [Eng07]. Figure 13 shows an example of PSSM where the view frustum is split into three partitions, and the shadow map for the middle partition map is highlighted. Using this approach, the sampling density decreases for each successive partition, because the same number of shadow map samples cover a larger and larger area.

In the most naive implementation, a PSSM scheme with n partitions requires n shadow rendering passes. Zhang et al. [ZSN07] describe three different methods to reduce the number of rendering passes. First, they evaluate the naive method that uses multiple passes for creating and rendering the shadow maps. This method does not require shaders and therefore runs on older hardware, but is also the slowest method on newer hardware and for complex scenes. The second method makes use of shaders and thereby avoids multiple passes for rendering. This gives a speed-up of up to

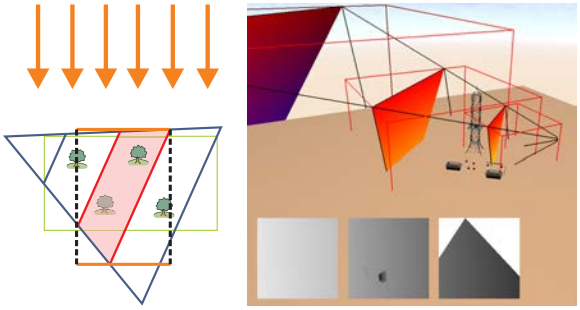

Figure 13: *PSSM:* Left: *The shadow map for the middle of three partitions of the view frustum (side view) is emphasized.* Right: *The bounding volumes for the partitions are shown in 3D. Inlays show the shadow maps.*

30% on GeForce 8800 and newer. The third method uses the geometry shader or advanced instancing to also avoid the multiply passes to create the shadow maps by replicating each triangle into each of the required shadow maps during the shadow rendering pass. Although the authors mention that this method further increases performance, no numerical data is given in their article to confirm this statement.

The most important question for this method is where to position the split planes. One way is to go back to the derivation of the shadow map resampling error. Each sub-shadow map could be interpreted as a big texel of a global shadow map, so that z-partitioning becomes a discretization of an arbitrary warping function. We have shown before that the optimal warping function is logarithmic, therefore the split positions C_i should be determined as [LTYM06]:

$$C_i = z_n \left(\frac{z_f}{z_n} \right)^{\frac{i}{m}}$$

where m is the number of partitions. However, as opposed to global warping schemes, the effect of z-partitioning is not limited to the axes of the shadow map, but even works when the light is directly behind the viewer (see Figure 14). This is the main advantage of z-partitioning over warping approaches, and the reason why z-partitioning is much more robust in general configurations. Figure 14, shows on the left the nearest and farthest partition in a situation with the light directly behind the viewer. The shadow map for the nearest partition covers a much smaller area, and therefore the perceived resolution is higher, just as is the case for the viewer projection. For instance, Tadamura et al. [TQJN99] and Engel [Eng07] partition the frustum along the view vector into geometrically increasing sub-volumes. Figure 15 shows a direct comparison of z-partitioning vs. warping in the case of a light from behind.

[ZSXL06] note that the optimal partition scheme is often not practical because it allocates most resolution near the near plane, which is rarely populated with objects.

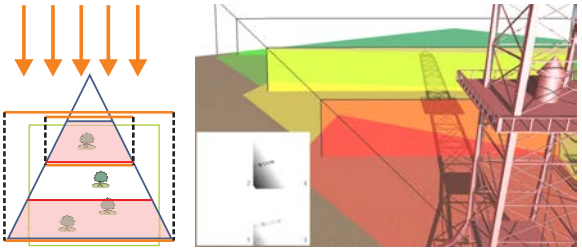

Figure 14: *PSSM even works for cases were warping fails: for instance when the light is coming from behind.*

They therefore propose computing the split positions as a weighted average between the logarithmic scheme and a simple equidistant split plane distribution. An alternative solution that better respects the theoretical properties of shadow map aliasing is to use a pseudo-near plane just as in warping. This approach is explained in Lloyd's thesis [Llo07] in Section 5.1.8.

[ZZB09] also discuss a number of practical issues related to z-partitioning, regarding flickering artifacts, shadow map storage strategies, split selection, computation of texture coordinates, and filtering across splits. An interesting observation is that in some cases, a point belonging to one partition should be shadowed using a shadow map generated for a different partition. This happens when the light is almost parallel to the view direction. In this case, the shadow maps for the partitions nearer the view point will provide better resolution.

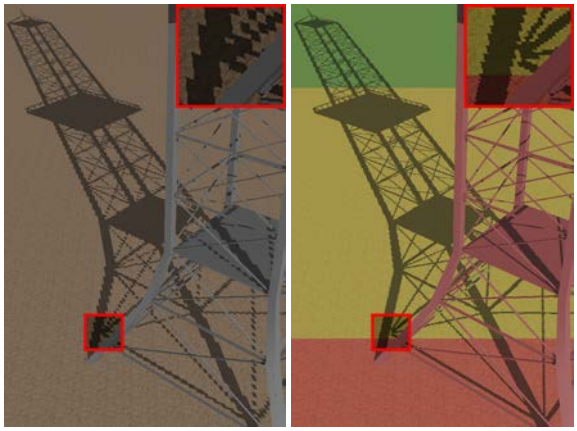

Figure 15: *For cases were the light is coming from behind, warping* (left) *gives unsatisfactory results, while z-partitioning* (right) *provides superior results. The shadow map used for each fragment is color coded.*

Still, optimizing the placement and sizes of z-partitions by hand can give superior results, especially if not the whole depth range is covered with shadows. To avoid this manual

tweaking, Lauritzen [Lau10] presented a probabilistic extension to cascaded shadow maps. The idea is to analyze the shadow sample distribution required by the current frame to find tight light-space bounds for each partition. The main limitation of this approach is that the clustering approach requires the calculation of a depth histogram, which is only feasible on the latest hardware.

4.3.2. Frustum Face Partitioning

Another alternative partitioning scheme is to use a separate shadow map for each face of the view frustum as projected onto the shadow map plane, and use warping for each shadow map separately. This can also be interpreted as putting a cube map around the post-perspective view frustum and applying a shadow map to each cube face [Koz04]. Each frustum face can be further split to increase quality.

This scheme is especially important because it can be shown that it is optimal for LogPSM, i.e., the combination of logarithmic and perspective shadow mapping introduced by Lloyd et al. [LGQ*08]. However, we will not elaborate this scheme here because Lloyd et al. [LGMM07] also showed that for practical situations, i.e., a large far plane to near plane ratio and a low number of shadow maps, z-partitioning (optionally combined with warping) is superior to frustum partitioning.

A split into different shadow map buffers involving a coarse scene analysis is described by Forsyth [For06]. The idea is that shadow receivers can be partitioned into different shadow maps according to their distance to the view point. An optimized projection can be used for each of these clusters, thereby only generating shadows were needed. This scheme can have advantages if shadows only occur sparsely in a scene, but for general settings (shadows everywhere) it is identical to z-partitioning (with the added overhead of the scene analysis).

4.3.3. Adaptive Partitioning

The advantage of the partitioning algorithms discussed so far is that they are very fast. On the other hand, they completely ignore surface orientation and therefore do not improve undersampling due to surfaces that are viewed almost edge-on by the light source, i.e., projection aliasing.

There are a number of algorithms that try to allocate samples in a more optimal way by analyzing the scene before creating the shadow map. This inevitably incurs some overhead due to the analysis step, but leads to much better results in general cases. This often necessitates a frame buffer read-back, which used to be a very costly step. Nevertheless, this cost has been reduced on recent hardware, which makes these methods more and more interesting for practical use. Prominent examples are *adaptive shadow maps* (ASM) [FFBG01, LSK*05], *resolution matched shadow maps* (RSMS) [LSO07], *queried virtual shadow maps* (QSM) [GW07b], *fitted virtual shadow maps* (FVSM) [GW07a], and *tiled shadow maps* (TiledSM) [Arv04].

All of these approaches rely on a hierarchical data structure (usually a quadtree) to refine the shadow map. They differ mainly in the termination criteria, and the measures that are required to determine this termination criterion.

The first approach to introduce adaptive partitioning for shadow maps are Fernando et al.'s *adaptive shadow maps* [FFBG01]. The idea is that a high-quality shadow map only needs high resolution at shadow edges. Therefore the shadow map is stored in a hierarchical grid structure (quad-tree). Each quad-tree node has a fixed resolution shadow map attached to it. Each frame the nodes can be split (creating new shadow maps for each split) iteratively to increase the shadow map resolution available. Lefohn et al. [LSK*05, LSO07] adapt this method by eliminating the edge detection phase in favor of generating all shadow map texels that are needed to resolve the shadowing of screen-space pixels (*resolution-matched shadow maps* (RMSM)). To make this approach feasible on a GPU, the authors use coherence between eye-space and light space: They assume that surfaces that are continuously visible in image space are also so in light space, and employ a connected-components analysis to find these surfaces and then request shadow map pages for each of those.

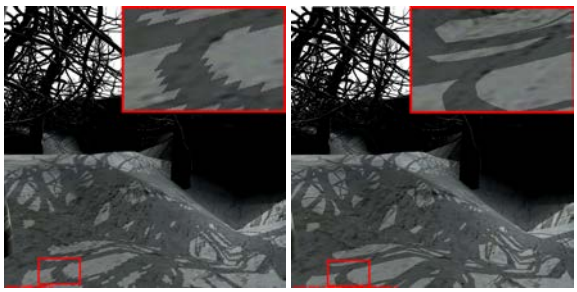

Figure 16: *Left: standard* 4096^2 *shadow map. Right: QVSM with a maximum refinement level of 32x32, and* 2048^2 *tiles.*

Queried Virtual Shadow Maps (QVSM), introduced by Giegl and Wimmer [GW07b], are maybe the adaptive partitioning scheme the easiest to implement, because they do not require a readback to compute the termination criterion, and do not require implementing hierarchical data structures on the GPU. The idea is very simple: refine a shadow map hierarchy until the actual change observed in the shadow due to a refinement lies below a predefined threshold. More exactly, starting from an initial shadow map (e.g., 2048x2048), this shadow map is split into 2x2 sub-tiles again with a resolution of 2048x2048 each. After each such refinement step, the scene is shadowed using the refined shadow maps, and the shadowing result is compared to the result of the previous step. If a certain threshold of changed pixels is exceeded

in a tile, refinement continues. The way to make this fast is to do all calculations on the GPU by using the occlusion query mechanism to count the number of pixels that differ when applying a more refined shadow map in comparison to the previous one. QVSM require a relatively high number of scene rendering passes, one for each refinement attempt. In order to avoid re-rendering the scene multiple times, the scene is rendered into a linear depth-buffer first and each rendering pass just uses this buffer to calculate shadows (also called deferred shadowing). Figure 16 shows a comparison of a large standard shadow map with QVSM.

In order to avoid the high number of shadow rendering passes in QVSMs, Giegl and Wimmer [GW07a] introduced Fitted Virtual Shadow Maps (FVSM) to try to determine beforehand what final refinement levels will be necessary in the quadtree. For this, the scene is rendered in a pre-pass, but instead of actually shadowing the scene, this pass just records the query location into the shadow map, as well as the required shadow map resolution at that query location. The resulting buffer is then transferred to the CPU. There, each sample of this buffer is transformed into shadow map space and stored in a low-resolution buffer, utilizing the efficient scattering capabilities of the CPU. This buffer ultimately contains the required resolution in each area of the shadow map, and the quadtree structure can be derived from it. In order to reduce penalties due to readback, only a small framebuffer (e.g., 256x256) is rendered in the pre-pass. In comparison to Adaptive Shadow Maps [FFBG01,LSK*05], both QVSM and FVSM are fast enough to evaluate the whole hierarchy for each frame anew and therefore work well for dynamic scenes, as opposed to ASM, which relies on an iterative edge-finding algorithm to determine refinement levels, and therefore needs to cache recently used tiles.

RMSM improve on ASMs especially for dynamic scenes, by avoiding the iterative step and calculating the required resolutions directly, somewhat similarly to FVSM. Both algorithms also mix data-parallel GPU algorithms [LKS*06] (like quadtree and sort) with standard rendering. In RMSMs, all steps are actually carried out on the GPU, while FVSM compute the required subdivision levels on the CPU, but on lower resolution buffers.

Tiled shadow maps [Arv04] tile the light view (here a fixed resolution shadow map) to change the sampling quality according to a heuristical analysis based on depth discontinuities, distances and other factors. This allows setting a hard memory limit, thereby trading speed against quality.

4.4. Comparison of Warping and Paritioning

Warping and partitioning are orthogonal approaches and can therefore be combined. For instance, for z-partitioning each partition can be rendered using LiSPSM. This increases quality especially for situations where LiSPSM works well (overhead lights). Figure 17 shows the effect of z-partitioning with and without warping.

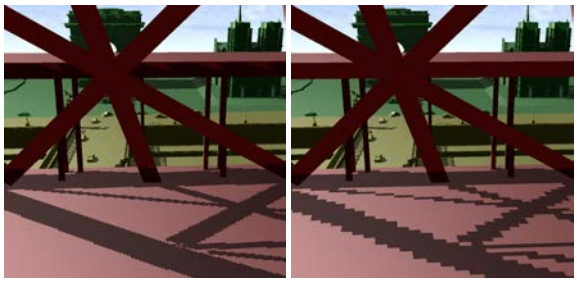

Figure 17: *Z-partitioning using 3 shadow maps with* (left) *and without* (right) *warping.*

One special case of such a combination is to use one uniform shadow map and one perspective shadow map and calculate a plane equation that separates areas where the one or the other provides the best quality [Mik07].

Figure 18 shows the overall error (here called storage factor), which takes into account error in both shadow map directions, of different schemes for different numbers of shadow maps for overhead lights (ideal for warping) and a light behind (no warping possible).

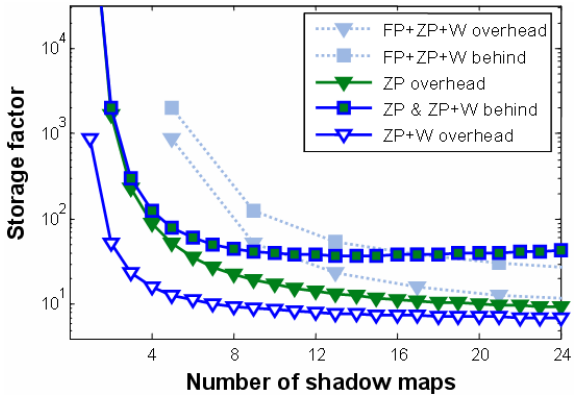

Figure 18: *Total error of different schemes for varying shadow map numbers. FP is frustum face partitioning, ZP is z-partitioning, W is warping (figure courtesy of Brandon Lloyd).*

4.5. Irregular Sampling

In the second pass of shadow mapping, all screen-space fragments are reprojected into the shadow map to be queried. The aliasing artifacts in hard shadow mapping stem from the fact that the shadow map query locations do not correspond to the shadow map sample locations (see Figure 19). Ideally, one would like to create shadow map samples exactly in those positions that will be queried later on. The idea of irregular sampling methods is to render an initial eye space pass to obtain the desired sample locations. These sample

locations are then used as pixel locations for the subsequent shadow map generation pass, thereby giving each screen-space fragment the best sample for the shadow map test and removing all aliasing artifacts. The challenge is that these new sample locations do not lie on a regular grid anymore. Therefore, view sample accurate shadow algorithms have to solve the problem of irregular rasterization.

Johnson at al. [JMB04, JLBM05] propose a hardware extension: they store a list of reprojected view samples at each regular shadow map grid element to allow for irregular sampling. They call this structure the irregular z-buffer. With this they can query the shadow test for each view sample by projecting each shadow casting triangle from the viewpoint of the light source. Each covered rasterized fragment has to be tested against each of the stored view samples and those in shadow are flagged. Unlike standard rasterization, which only creates a sample if the location at the center of a fragment is inside the rasterized polygon, this approach needs to calculate the overlap of each triangle with each fragment. This is necessary because the eye space samples are located at arbitrary positions inside each grid element. Finally, a screen-space quad is rendered in eye-space, where each fragment does a shadow query by testing its corresponding list entry for the shadow flag.

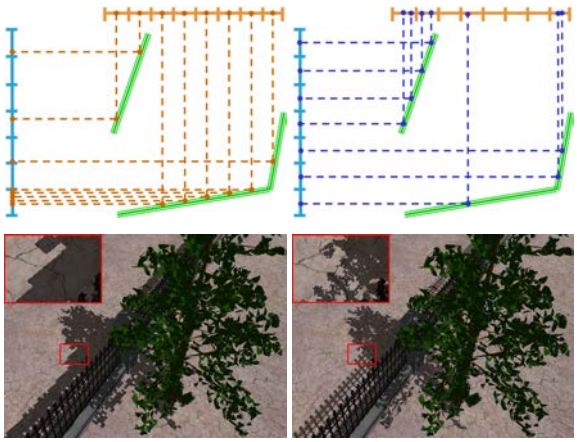

Figure 19: *Samples created on a regular grid in the shadow map can be irregular in eye space* (upper-left) *and vice versa* (upper-right)*. Therefore regular shadow mapping can lead to undersampling artifacts* (lower-left) *while irregular shadow mapping avoids artifacts by sampling the shadow map for each eye space sample* (lower-right).

Alias-free shadow maps [AL04] provide a hierarchical software implementation using an axis-aligned BSP tree to efficiently evaluate shadow information at the required sample points. This approach was later mapped to graphics hardware by Arvo [Arv07] and Sintorn et al. [SEA08]. This approach is very similar to Johnson et al.'s, but does not require any hardware changes because the list stored at each shadow map element is realized with a constant memory footprint. They are also able to map the overlap calculation to hardware by using conservative rasterization. The method is suited to be combined with reparametrization methods and in practice, the authors implemented a variant of the fitting approach described in [BAS02a]. Even accurate per-pixel shadows can be improved further by introducing supersampling. Pan et al. [PWC*09] present such a method, which avoids brute-force supersampling by extending [SEA08]. The main idea is to approximate pixels by small 3d quadrangles called facets (instead of just points). These facets allow accounting for the area of a pixel. Potential blockers are projected into screen-space via facets. Here occlusion masks with multiple samples per pixel are used to calculate the subpixel shadows.

4.6. Temporal Reprojection

Finally, one way to increase the sampling rate is by reusing samples from previous frames through reprojection [SJW07]. The main idea is to jitter the shadow map viewport differently in each frame and to combine the results over several frames, leading to a much higher effective resolution.

This method requires an additional buffer to store the accumulated history of previous frames. In the current frame, the result of the shadow map lookup is combined with the accumulated result calculated in the previous frames, which can be looked up using reprojection (to account for movement). If a depth discontinuity between the new and the reprojected sample is detected, then the old result is discarded since it is probably due to a disocclusion.

Figure 20: *LiSPSM gives good results for a shadow map resolution of* 1024^2 *and a viewport of* 1680×1050*, but temporal reprojection can give even better results because it is not limited by the shadow map resolution.*

The shadow quality in this approach can actually be made to converge to a pixel-perfect result by optimizing the choice of the weight between the current and the previous frame result (see Figure 20). The weight is determined according to the *confidence* of the shadow lookup:

$$\text{conf}_{x,y} = 1 - \max\left(|x - \text{center}_x|, |y - \text{center}_y|\right) \cdot 2,$$

where $\text{conf}_{x,y}$ is the confidence for a fragment projected to

(x, y) in the shadow map and $(\text{center}_x, \text{center}_y)$ is the corresponding shadow map texel center.

The confidence is higher if the lookup falls near the center of a shadow map texel, since only near the center of shadow map texels it is very likely that the sample actually represents the scene geometry.

Note that reprojection based approaches take a few frames to converge after quick motions. Also, they cannot deal very well with moving objects or moving light sources. On the other hand, they practically eliminate temporal aliasing, for example due to shadow map focusing.

5. Depth Biasing

A problem that is known by the name of *incorrect self-shadowing* or *shadow acne* is caused by undersampling and the imprecision of the depth information stored in the shadow map for each texel. On the one hand, depth is represented with limited precision using either a fixed or floating-point representation and these imprecisions can lead to wrong results. And on the other hand, the depth of each reprojected view space fragment is compared to a single depth from the shadow map. This depth is only correct at the original sampling point, but is used for the whole texel area. If this texel area is big in view-space, due to undersampling, incorrect shadow test outputs can be the result (see Figure 21).

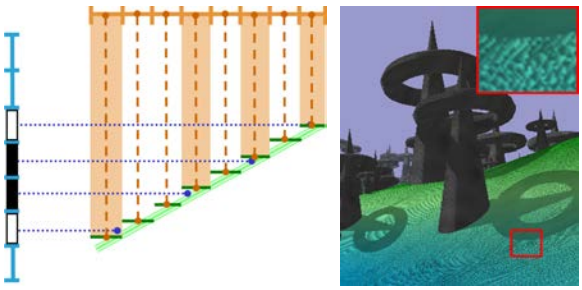

Figure 21: Left: *A polygon is shadowing itself because of insufficient sampling and depth precision in the shadow map.* Right: *This results in z-fighting.*

The standard solution is a user-defined depth bias, a small increment added to the shadow map depth values to move them further away (see Figure 22, left). This moves the shadow caster away from the light and can therefore introduce light leaks if the new depth is farther away than the depth of the receiver. This is most noticeable for contact shadows (see Figure 22, right). To make one bias setting applicable to a wider range of geometry, most implementations provide a second parameter, which is dependent on the polygon slope *(slope-scale biasing)*. Nevertheless, depth biasing is highly scene dependent and in general no automatic solution for an arbitrary scene exists. The main benefit of this method is its simplicity and support through hardware.

A factor that further aggravates the precision issues is the non-linear distribution of depth values introduced by point (spot) lights, PSM, TSM, LispSM and similar reparameterization methods. This non-linear distribution of depth values is generated by the perspective transformation that involves a $1/w$ term, generating a hyperbolic depth value distribution. To counteract this, Brabec et al. [BAS02a] proposed linearly distributed depth values. A similar approach was chosen for trapezoidal shadow maps (TSM [MT04]), where the authors recommend omitting the z-coordinate from the perspective transformation. Kozlov [Koz04] proposes to use slope-scale biasing for PSM in world-space and later transforms the results into post-projective space. LispSM has less problems with self-shadowing artifacts and can use normal slope-scale biasing.

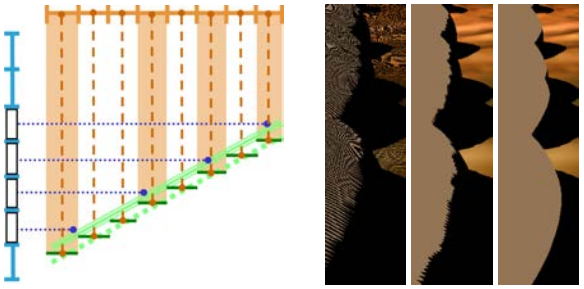

Figure 22: *Depth biasing can remove incorrect self-shadowing* (left)*, but can also introduce light leaks* (right) *if chosen too big.*

To remove depth biasing, Woo [Woo92] proposed calculating the average of the first and second depth surface and consequently using this average depth (termed *midpoint shadow map*) for the depth comparison. This introduces the overhead of some form of depth peeling to acquire the second depth layer.

Second-depth shadow mapping, as proposed by Wang and Molnar [WM94], builds on the simple idea of using only the depth of the second nearest surface to the light source, which can be done efficiently by backside rendering if shadow casters are solid objects. The shadow map depth comparison is therefore shifted to the back side of the casting geometry, making the shadow test more robust for polygons facing the light source. In essence this introduces an adaptive depth bias with the size of the thickness of the shadow caster. This also introduces light leaks on shadow casting backsides. Fortunately, these backsides can be determined to be in shadow anyway by application of a standard diffuse illumination model. Nevertheless, due to possible huge differences in nearest and second nearest surface (huge shadow caster thickness), imprecisions can arise. This problem is addressed by Weiskopf and Ertl in *dual shadow maps* [WE03]. They reintroduce a parameter that in effect limits the shadow caster thickness.

A method that can avoid the need for biasing altogether was introduced by Hourcade and Nicolas [HN85]: a unique polygon id is stored instead of the depth in the shadow map. On comparison, either the same id is found (lit) or different ids are present (in shadow). To store the unique id, one 8bit channel (256 ids) will be insufficient in most cases. Because only one id can be stored per texel, the mechanism breaks down if more than one triangle is present per texel.

6. Strategies to Reduce Reconstruction and Oversampling Errors

The standard shadow map test results are binary: Either a fragment is in shadow or not, creating hard jagged shadow map edges for undersampled portions of the shadow map. From a signal processing point of view this corresponds to *reconstruction* using a box filter. Traditional bilinear reconstruction as used for color textures is inappropriate for shadow maps, because a depth comparison to an interpolated depth value still gives a (even more incorrect) binary result instead of an improved reconstruction.

Figure 23: *Undersampled unfiltered shadow maps on the* left *suffer from hard jagged edges. These can be removed by filtering. On the* right *hardware* PCF *with a 2x2 kernel is applied.*

Reeves et al. [RSC87] discovered that it makes much more sense to reconstruct the shadow test *results* and not the original depth values. His *percentage closer filtering (PCF)* technique averages the results of multiple depth comparisons in a Poisson disk sampling pattern in the shadow map to obtain in essence a (higher order) reconstruction filter for magnification of the shadow map. The smoothness of the resulting shadow is directly related to the filter kernel size. Note that this kernel has to be evaluated for each view space fragment, making the algorithm's performance highly sensitive to the kernel size. A faster variation of this method, already implemented directly in the texture samplers of current hardware, is to bilinearly filter the shadow map test results of the four neighboring shadow map samples (see Figure 23).

Aliasing due to *oversampling* is usually avoided in image processing by band-limiting the reconstructed signal before resampling it at the final pixel locations. For texture mapping, prefiltering approaches such as mip-mapping are most common. However, this is much harder to do for shadow mapping since the shadow function is not a linear transformation of the depth map, and therefore the bandlimiting step cannot be done before rendering. One option is to resort to on-the-fly filtering and evaluate PCF with large filter kernels, however this is slow and does not scale. Recent research proposed clever ways to reformulate the shadow test into a linear function so that prefiltering can be applied.

One such reformulation are *variance shadow maps* [DL06, Lau07] introduced by Donnelly and Lauritzen. They estimate the outcome of the depth test for a given PCF kernel by using mean and variance of the depth value distribution inside this kernel window. The advantage is that mean and variance can be precomputed using for example mip-mapping. The problem with this approach is that high variance in the depth distributions (high depth complexity) can lead to light leak artifacts (see Figure 24, left) and high-precision (32 bit floating point) texture filtering hardware is needed for satisfying results. A solution to both problems *(layered variance shadow maps)* was presented by Lauritzen and McCool [LM08], who partition the depth range of the shadow map into multiple layers. Although texture precision can be reduced with this approach (even down to 8 bit), multiple layers are still required for low variance.

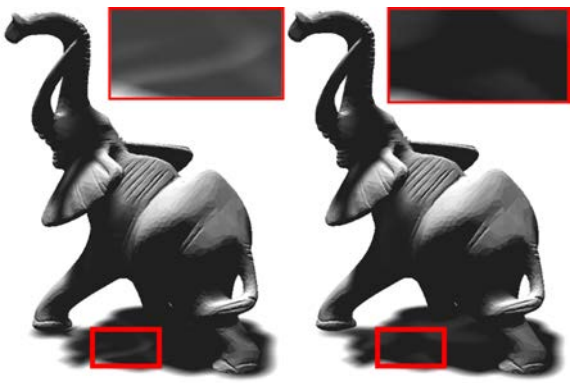

Figure 24: *Variance shadow maps* (left)*, in contrast to convolution shadow maps, suffer* (right) *from light leaks.*

Another way to reformulate the binary shadow test was introduced by Annen et al. with *convolution shadow maps* [AMB*07] (see Figure 24, right). Here instead of statistical estimate, a Fourier expansion is used to represent the depth test. For a practical approximation using 16 coefficients (from the infinitely many), 16 sine and 16 cosine textures have to be stored. The expansion into a Fourier basis is a linear operation, so that prefiltering using mip-mapping of the individual basis textures can be applied. While the basis textures require only 8 bit per texel (in comparison to 24 bit for standard shadow maps), memory considerations still require a restriction of the Fourier expansion to a small

number of terms, which introduces ringing artifacts (Gibb's phenomenon), again resulting in light leaks.

Following the same general idea, Annen et al. [AMS*08] proposed *exponential shadow maps*, which replace the Fourier expansion by an exponential. The idea is to interpret the shadow test as a step function and use the exponential as a separable approximation of this step function. Here a single 32 bit texture channel is sufficient, making this approach much more memory friendly, but for larger kernel sizes this approximation does not hold anymore, leading to artifacts.

Better reconstruction can also be achieved by changing the reconstruction algorithm itself. *Shadow silhouette maps* [SCH03,Sen04], for example, allow reconstructing linear shadow boundaries by additionally storing a point on the silhouette for each shadow map texel. For reconstruction of the shadow caster edges, the silhouette points of neighboring texels are evaluated. Artifacts are visible if more than one silhouette is crossing the texel area, so the approach is still heavily dependent on the resolution of the shadow map. The performance is mainly limited by the costly silhouette point determination.

Figure 25: *Deep shadow maps store a piecewise linear representation of the transmittance function gathered from various samples at every texel* (left). *This allows shadow mapping of challenging cases like hair* (right).

A very sophisticated off-line filtering approach are *deep shadow maps* [LV00]. Here each texel contains a compressed piecewise linear representation of the visibility function – a weighted average of n piecewise linear transmittance functions taken at samples on the texel's area. This representation can be prefiltered and allows high quality shadows for complex cases such as hair or clouds (see Figure 25). Hadwiger et al. [HKSB06] presented an interactive version for volume ray-casting on the GPU.

7. Conclusion

Finally, we give some practical hints which algorithms to use in what situation.

If the requirement is that only a single shadow map should be used, i.e., the algorithm should run at the same speed as standard shadow mapping, then *light space perspective shadow mapping*, with the modification by Lloyd et al., is the best algorithm. This algorithm will achieve excellent quality in many configurations, especially in outdoor scenarios with roughly overhead lighting, however it can easily degrade to the quality of (focused) uniform shadow mapping. With the modification by Lloyd et al., it will never degrade below the quality of uniform shadow mapping.

If more than one shadow map is allowed, i.e., some performance loss can be accepted, the best known tradeoff between efficiency and quality is achieved by *z-partitioning* (CSM, PSSM). The distribution of multiple shadow maps mimics a very rough approximation of the optimal logarithmic shadow map reparametrization. Furthermore, each shadow map can adapt optimally to one part of the view frustum, thus improving the quality in each spatial dimension, independent of the orientation of the view frustum. It is possible to combine z-partitioning with a reparametrization, however, temporal aliasing is increased by this approach, and the gain is not very high.

One major advantage of the aforementioned algorithms is that they are scene-independent, and thus do not require interaction (e.g., readback) with the scene. On the other hand, this limits these approaches to dealing with perspective aliasing only, while local aliasing effects due to different surface orientations, causing projection aliasing, cannot be improved. If higher quality is desired, then adaptive partitioning algorithms should be applied. In the future, even irregular sampling approaches, which really result in a pixel-accurate solution, might become feasible for real world applications. For the special case of a static scene with a static light source, *temporal reprojection* is a powerful method that gives high-quality shadows.

To fight shadow acne, *backside rendering* is the fastest way to go. This moves most acne for solid objects to the backside. Here either the light model is chosen to darken this areas further, making the remaining artifacts inconspicuous, or an additional bias removes acne also in these areas. However, this is no robust solution for thin/non solid objects.

For filtering with small filter kernels, PCF (especially in hardware) is fast and can remove some of the reconstruction errors. For larger filter kernels PCF is too slow. Currently, *layered variance shadow maps* are the fastest and most robust solution for this case.

References

[AL04] AILA T., LAINE S.: Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004* (Norrköping, Sweden, 2004), Eurographics Association, pp. 161–166. 12

[AMB*07] ANNEN T., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Convolution shadow maps. In *Rendering Techniques 2007: Eurographics Symposium on Rendering* (Grenoble, France, June 2007), Kautz J., Pattanaik S., (Eds.), vol. 18 of *Eurographics / ACM SIGGRAPH Symposium Proceedings*, Eurographics, pp. 51–60. 14

[AMS*08] ANNEN T., MERTENS T., SEIDEL H.-P., FLERACKERS E., KAUTZ J.: Exponential shadow maps. In *GI '08: Proceedings of graphics interface 2008* (Toronto, Ont., Canada, Canada, 2008), Canadian Information Processing Society, pp. 155–161. 15

[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. 10, 11

[Arv07] ARVO J.: Alias-free shadow maps using graphics hardware. *Journal of graphics, gpu, and game tools 12*, 1 (2007), 47–59. 12

[BAS02a] BRABEC S., ANNEN T., SEIDEL H.-P.: Practical shadow mapping. *Journal of Graphics Tools: JGT 7*, 4 (2002), 9–18. 5, 12, 13

[BAS02b] BRABEC S., ANNEN T., SEIDEL H.-P.: Shadow mapping for hemispherical and omnidirectional light sources. In *Advances in Modelling, Animation and Rendering (Proceedings Computer Graphics International 2002)* (Bradford, UK, 2002), Vince J., Earnshaw R., (Eds.), Springer, pp. 397–408. 2

[CG04] CHONG H., GORTLER S. J.: A lixel for every pixel. In *Proceedings of Eurographics Symposium on Rendering 2004* (Norrköping, Sweden, 2004). 6

[CG07] CHONG H. Y., GORTLER S. J.: *Scene Optimized Shadow Mapping. Harvard Computer Science Technical Report: TR-07-07*. Tech. rep., Harvard University, Cambridge, MA, 2007. 6

[Cho03] CHONG H.: *Real-Time Perspective Optimal Shadow Maps*. Senior thesis, Harvard College, Cambridge, Massachusetts, Apr. 2003. 6

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, July 1977), George J., (Ed.), vol. 11, ACM Press, pp. 242–248. 2

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 161–165. 14

[Eng07] ENGEL W.: Cascaded shadow maps. In *ShaderX^5: Advanced Rendering Techniques*. Charles River Media, Inc., 2007. 8, 9

[FFBG01] FERNANDO R., FERNANDEZ S., BALA K., GREENBERG D. P.: Adaptive shadow maps. In *SIGGRAPH 2001 Conference Proceedings* (Los Angeles, USA, Aug. 2001), Fiume E., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 387–390. 10, 11

[For06] FORSYTH T.: *Making shadow buffers robust using multiple dynamic Shadow Maps*. Charles River Media, 2006. 10

[GHFP08] GASCUEL J.-D., HOLZSCHUCH N., FOURNIER G., PÉROCHE B.: Fast non-linear projections using graphics hardware. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games* (Redwood City, California, 2008), ACM, pp. 107–114. 2

[GW07a] GIEGL M., WIMMER M.: Fitted virtual shadow maps. In *Proceedings of GI (Graphics Interface)* (Montreal, Canada, May 2007), Healey C. G., Lank E., (Eds.), Canadian Human-Computer Communications Society, pp. 159–168. 10, 11

[GW07b] GIEGL M., WIMMER M.: Queried virtual shadow maps. In *Proceedings of I3D (ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games)* (Seattle, WA, Apr. 2007), ACM Press, pp. 65–72. 10

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: Gpu-accelerated deep shadow maps for direct volume rendering. In *GH '06: Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware* (New York, NY, USA, 2006), ACM, pp. 49–52. 15

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. *Computer Graphics Forum 22*, 4 (dec 2003), 753–774. 1

[HN85] HOURCADE J. C., NICOLAS A.: Algorithms for antialiased cast shadows. *Computers and Graphics 9*, 3 (1985), 259–265. 14

[JLBM05] JOHNSON G. S., LEE J., BURNS C. A., MARK W. R.: The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Trans. Graph. 24*, 4 (2005), 1462–1482. 12

[JMB04] JOHNSON G. S., MARK W. R., BURNS C. A.: *The Irregular Z-Buffer and its Application to Shadow Mapping*. Research paper, The University of Texas at Austin, 2004. 12

[Koz04] KOZLOV S.: Perspective shadow maps - care and feeding. *GPU Gems 1* (2004), 217–244. 10, 13

[Lau07] LAURITZEN A.: *GPU Gems 3*. Addison-Wesley, 2007, ch. Summed-Area Variance Shadow Maps. 14

[Lau10] LAURITZEN A.: Sample distribution shadow maps. In *Advances in Real-Time Rendering in 3D Graphics and Games* (2010), SIGGRAPH Courses. 10

[LGMM07] LLOYD D. B., GOVINDARAJU N. K., MOLNAR S. E., MANOCHA D.: Practical logarithmic rasterization for low-error shadow maps. In *Proceedings of Graphics Hardware (ACM SIGGRAPH/Eurographics Workshop on GH)* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 17–24. 7, 10

[LGQ*08] LLOYD D. B., GOVINDARAJU N. K., QUAMMEN C., MOLNAR S. E., MANOCHA D.: Logarithmic perspective shadow maps. *ACM Transactions on Graphics 27*, 4 (Oct. 2008), 1–32. 4, 7, 10

[LKS*06] LEFOHN A., KNISS J. M., STRZODKA R., SENGUPTA S., OWENS J. D.: Glift: Generic, efficient, random-access GPU data structures. *ACM Transactions on Graphics 25*, 1 (Jan. 2006), 60–99. 11

[Llo07] LLOYD B.: *Logarithmic Perspective Shadow Maps*. PhD thesis, University of North Carolina at Chapel Hill, August 2007. 4, 7, 8, 9

[LM08] LAURITZEN A., MCCOOL M.: Layered variance shadow maps. In *GI '08: Proceedings of graphics interface 2008* (Toronto, Ont., Canada, Canada, 2008), Canadian Information Processing Society, pp. 139–146. 14

[LSK*05] LEFOHN A., SENGUPTA S., KNISS J. M., STRZODKA R., OWENS J. D.: Dynamic adaptive shadow maps on graphics hardware. In *ACM SIGGRAPH Conference Abstracts and Applications* (Los Angeles, CA, Aug. 2005). 10, 11

[LSO07] LEFOHN A. E., SENGUPTA S., OWENS J. D.: Resolution matched shadow maps. *ACM Transactions on Graphics 26*, 4 (Oct. 2007), 20:1–20:17. 10

[LTYM06] LLOYD B., TUFT D., YOON S., MANOCHA D.: Warping and partitioning for low error shadow maps. In *Proceedings of the Eurographics Symposium on Rendering 2006* (2006), Eurographics Association, pp. 215–226. 6, 8, 9

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 385–392. 15

[MBW08] MATTAUSCH O., BITTNER J., WIMMER M.: Chc++: Coherent hierarchical culling revisited. *Computer Graphics Forum (Proceedings Eurographics 2008) 27*, 2 (Apr. 2008), 221–230. 5

[Mik07] MIKKELSEN M. S.: Separating-plane perspective shadow mapping. *journal of graphics, gpu, and game tools 12*, 3 (2007), 43–54. 11

[MT04] MARTIN T., TAN T.-S.: Anti-aliasing and continuity with trapezoidal shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004* (Norrköping, Sweden, 2004), pp. 153–160. 6, 13

[PWC*09] PAN M., WANG R., CHEN W., ZHOU K., BAO H.: Fast, sub-pixel antialiased shadow maps. *Computer Graphics Forum 28*, 7 (Oct. 2009), 1927–1934. 12

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), Stone M. C., (Ed.), vol. 21, pp. 283–291. 14

[SCH03] SEN P., CAMMARANO M., HANRAHAN P.: Shadow silhouette maps. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 22*, 3 (2003), 521–526. 15

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (San Antonio, Texas, July 2002), ACM Press, pp. 557–562. 3, 4, 6

[SEA08] SINTORN E., EISEMANN E., ASSARSSON U.: Sample-based visibility for soft shadows using alias-free shadow maps. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2008) 27*, 4 (June 2008), 1285–1292. 12

[Sen04] SEN P.: Silhouette maps for improved texture magnification. In *Proceedings of Graphics Hardware (ACM SIGGRAPH/Eurographics Workshop on GH)* (Grenoble, France, 2004), ACM, pp. 65–73. 15

[SJW07] SCHERZER D., JESCHKE S., WIMMER M.: Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)* (Grenoble, France, June 2007), Kautz J., Pattanaik S., (Eds.), Eurographics, Eurographics Association, pp. 45–50. 12

[TQJN99] TADAMURA K., QIN X., JIAO G., NAKAMAE E.: Rendering optimal solar shadows using plural sunlight depth buffers. In *CGI '99: Proceedings of the International Conference on Computer Graphics* (Washington, DC, USA, 1999), IEEE Computer Society, p. 166. 8, 9

[WE03] WEISKOPF D., ERTL T.: Shadow Mapping Based on Dual Depth Layers. In *Procceedings of Eurographics '03 Short Papers* (Granada, Spain, 2003), pp. 53–60. 13

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings) 12*, 3 (Aug. 1978), 270–274. 2

[WM94] WANG Y., MOLNAR S.: *Second-Depth Shadow Mapping*. Tech. rep., University of North Carolina at Chapel Hill, 1994. 13

[Woo92] WOO A.: The shadow depth map revisited. *Graphics Gems III* (1992), 338–342. 13

[WPF90] WOO A., POULIN P., FOURNIER A.: A survey of shadow algorithms. *IEEE Computer Graphics and Applications 10*, 6 (Nov. 1990), 13–32. 1

[WS06] WIMMER M., SCHERZER D.: Robust shadow mapping with light space perspective shadow maps. In *ShaderX 4 – Advanced Rendering Techniques*, Engel W., (Ed.), vol. 4 of *ShaderX*. Charles River Media, Mar. 2006. 5

[WSP04] WIMMER M., SCHERZER D., PURGATHOFER W.: Light space perspective shadow maps. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)* (Norrköping, Sweden, 2004), Springer Computer Science, Eurographics, Eurographics Association. 5, 6, 7

[ZSN07] ZHANG F., SUN H., NYMAN O.: Parallel-split shadow maps on programmable gpus. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley, Aug. 2007. 8

[ZSXL06] ZHANG F., SUN H., XU L., LUN L. K.: Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications* (Hong Kong, China, 2006), VRCIA '06, ACM, pp. 311–318. 8, 9

[ZXTS06] ZHANG F., XU L., TAO C., SUN H.: Generalized linear perspective shadow map reparameterization. In *Proceedings of VRCIA (International Conference on Virtual Reality Continuum and Its Applications)* (Hong Kong, China, 2006), ACM, pp. 339–342. 7

[ZZB09] ZHANG F., ZAPRIJAGAEV A., BENTHAM A.: Practical cascaded shadow maps. In *ShaderX 7 – Advanced Rendering Techniques*, Engel W., (Ed.), vol. 7 of *ShaderX*. Charles River Media, Mar. 2009. 6, 9