Improved Real-Time Shadow Mapping for CAD Models

Vitor Barata R. B. Barroso
TecGraf, Computer Science Dept., PUC-Rio
vbarata@tecgraf.puc-rio.br

Waldemar Celes
TecGraf, Computer Science Dept., PUC-Rio
celes@tecgraf.puc-rio.br

Abstract

Shadow mapping is a widely used rendering technique for real-time shadow generation. However, it may produce jagged shadow borders and incorrect self-shadowing artifacts. When applied to CAD (Computer-Aided Design) models, it presents additional challenges due to the existence of thin and complex-silhouette objects, high depth range, and high depth complexity. In this work, we present a short survey of shadow mapping algorithms and investigate their suitability to CAD model rendering. We also present two improvements to existing techniques: a generalized parameter for LiSPSMs (Light-Space Perspective Shadow Maps) and an adaptive z-partitioning scheme.

1. Introduction

Shadows are very important visual effects on computer-generated scenes because they reveal information about the spatial relationships among objects and enhance the realism of the visualization. Unfortunately, generating fast and accurate shadows is still challenging due to the global nature of the problem.

Recently, there has been a lot of discussion about shadow mapping, a widely used shadow rendering technique due to its simplicity, efficiency and generality. The technique is particularly attractive to CAD model rendering because it does not require an expensive analysis of the model's geometry (although geometry shaders are making such analysis cheaper).

As an image-space algorithm, shadow mapping [1] suffers from aliasing problems due to the limited resolution available for sampling the scene. This results in jagged shadow borders and the incorrect self-shadowing of objects. These artifacts are particularly strong and difficult to avoid in CAD models because of their high depth range and depth complexity, and the presence of many thin complex-silhouette objects such as bars and lattices.

In this paper, we present a survey of shadow mapping techniques that address the problems of self-shadowing and aliasing. Each algorithm is analyzed considering visual quality, computational efficiency, ease of implementation and suitability to CAD model rendering. Additionally, we present two improvements to existing techniques. First, we build on the works of Wimmer et al. [2] and Zhang et al. [3] to derive a LiSPSM (Light-Space Perspective Shadow Map) parameter generalized to different angles between the light and view directions. Second, we propose an adaptive z-partitioning scheme based on the work of Lloyd et al. [4,5]. By combining these improvements with a set of previous techniques, we achieved real-time high-quality shadow rendering, as illustrated in Figure 1.

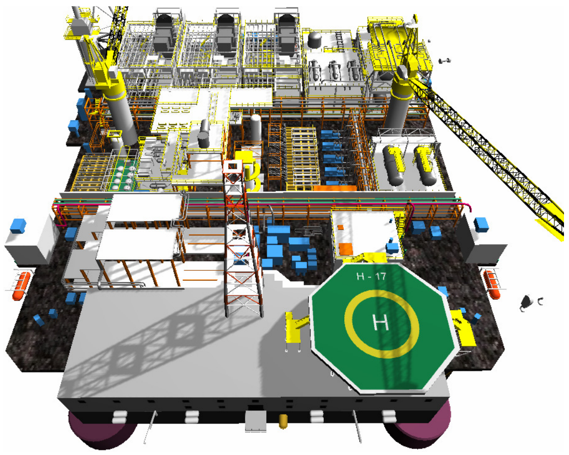


Figure 1. Shadow-mapped CAD Model

The remainder of the paper is organized as follows: Section 2 reviews and analyses several existing techniques and their suitability to CAD model rendering. Section 3 presents our generalized LiSPSM parameter and our adaptive z-partitioning scheme. Section 4 presents results and additional remarks. Finally, conclusions are drawn in Section 5.

2. Survey of shadow mapping techniques

The standard shadow mapping technique [1] first renders the scene from the viewpoint of the light source and stores the depth buffer in a texture called shadow map. Then, the scene is rendered again from the viewer's position. Each pixel is transformed back into the light space, where it is determined to be in shadow if its depth is farther from the light source than the corresponding value in the shadow map.

The main problems associated with shadow mapping are incorrect self-shadowing and aliasing artifacts. In this section, we analyze existing techniques that address these problems and discuss their suitability for CAD model rendering.

2.1. Depth bias and sample alignment

When using the shadow mapping technique, incorrect self-shadowing artifacts may appear because of the limited precision and resolution available to sample depth values throughout the scene.

Ideally, the depth of a lit pixel in relation to the light source is equal to the corresponding value stored in the shadow map. However, when a pixel is transformed into light space, it does not align exactly with the center of a shadow map texel [6]. Therefore, the two samples do not actually represent the same point, which produces wrong shadow test results: a lit pixel may be considered to be in shadow.

Usually, the self-shadowing problem is countered by adding a small bias to the depth values stored in the shadow map [1]. However, if the bias is exaggerated, shadows are visibly pushed away from their true positions. The choice of an appropriate value can be hard, since it depends on factors such as the thickness of scene objects, the relative position of the camera and the light source, and the shadow map resolution [6].

A generic bias z_{bias} can be conveniently expressed as a function of the depths of the first (z_1) and second (z_2) surfaces nearest to the light source [8]. The value stored in the shadow map is then given by:

$$z_{map} = z_1 + z_{bias}(z_1, z_2) \tag{1}$$

Table 1 summarizes the z_{bias} functions proposed by different authors.

The Second-Depth technique, (b) in Table 1, requires the scene to include only thick solid objects. When this condition is satisfied, the bias can be implemented easily and efficiently with polygon front-face culling, and the overall result is often satisfactory. Unfortunately, however, CAD models often include non-solid objects, which invalidate this approach.

Midpoint-based techniques, (c) and (d) in Table 1, disregard back-facing surfaces for z_2. The upper bound in (d) is required to avoid self-unshadowing artifacts when using back-face culling [8]. Although these techniques produce the best results in a general scene, they have the drawback of requiring an additional shadow map generation pass to obtain the second surface using depth peeling. Therefore, they are more recommended

Table 1. Typical z_{bias} functions

Technique	$z_{bias}(z_1,z_2)$
(a) Constant Bias [1]	z_{offset}
(b) Second-Depth [6]	$z_2 - z_1$
(c) Midpoint [7]	$(z_2 - z_1)/2$
(d) Dual-Depth [8]	$\min((z_2 - z_1)/2, z_{max})$

for static scenes and uniform shadow maps (with no view-dependent warping or partitioning), which may be generated only once. However, since warping is highly recommended for CAD model visualization (Section 2.3), the additional cost of midpoint techniques may become prohibitive. Because of that, we generally recommend the use of a simple constant bias.

As a different approach to avoid self-shadowing artifacts, Wang & Molnar [6] introduce the technique of sample alignment. Since the camera samples do not lie on the centers of the shadow map texels, they calculate a virtual sample, on the plane tangent to the object surface, which does lie in the right spot. Note that the alternative, reconstructing the shadow map value that matches each camera sample, does not work well, since interpolated depth values between two different occluders have no useful meaning.

When combined with depth bias, camera sample alignment greatly reduces self-shadowing artifacts (although some issues persist, as discussed in [6]). Therefore, the technique is recommended when depth bias alone cannot avoid the problem.

2.2. Filtering

Jagged shadow borders are usually filtered using PCF (Percentage-Closer Filter) [9], which is applied to the results of multiple shadow tests. A more efficient, recently presented alternative is the use of VSMs (Variance Shadow Maps) [10].

With VSMs, the shadow map stores not only the depth, but also the squared depth of each sample. These values are filtered and then used to determine the mean μ and variance σ^2 of all the samples inside the filter kernel. In the final rendering pass, Chebychev's inequality is calculated for every pixel:

$$P(z_{map} \geq z_{pixel}) \leq \frac{\sigma^2}{\sigma^2 + (z_{pixel} - \mu)^2} \equiv p_{\max} \tag{2}$$

In Equation (2), p_{max} is the maximum probability that a pixel should be lit, considering the distribution of depths inside the filter region. This value is used directly as the resulting light intensity for the pixel.

Unfortunately, VSMs are not well suited for CAD model rendering. In their paper, Donnely & Lauritzen pointed out that VSMs may produce light bleeding artifacts [10]. This effect is particularly strong in CAD models due to the high depth range and depth complexity, as can be seen in Figure 2. In region A, the silhouette of a tower can be seen because of a high σ^2 value. In region B, light bleeding occurs because the filtered depth values (μ) are interpolated with a far away object, inverting the result of the depth test and skipping the calculation of p_{max} according to the VSM algorithm. Finally, in region C, the term $(z_{pixel} - \mu)$ in (2) is very small, making p_{max} rise and the shadow disappear. Because of such strong light bleeding artifacts, VSMs are not recommended for CAD model visualization, and PCF should be used instead. Current researches on "Summed Area VSMs" [11] have attempted to eliminate light bleeding, but such an approach has yet to be tested.

Percentage-closer filtering is already supported directly by some video cards with a 2×2 kernel. Larger filters are also becoming less expensive with the increasing fragment processing power of current graphics hardware. Kryachko's implementation [12] of a circular filter kernel with jittered samples seems very appealing, since it efficiently eliminates banding effects. On the other hand, the test samples and adaptive refinement proposed in that work may introduce artifacts in shadows of complex-silhouette objects such as lattices, and therefore this optimization should be avoided when performance is not at critical levels. We recommend a 3×3 to 5×5 kernel size.

One aspect of PCF is often overlooked. The kernel should be considered in texel space, not in pixel space. In other words, the distance between the centers of two cells should be that of one texel. This means the filter region will automatically adjust its size to blur the aliased shadow border region. However, shadows with no visible aliasing will remain unblurred as hard-shadows. Unfortunately, this is necessary to guarantee proper blur where aliasing is strong. After all, PCF is an antialiasing technique, not properly a soft shadow one.

Normally, PCF's shadow tests compare only one camera sample with several shadow map values, which aggravates self-shadowing artifacts. In order to avoid this problem, multiple camera virtual samples should be generated on the surface's tangent plane [6]. When a regular filter kernel is used in texel space, only the first virtual sample must be calculated, while the others can be obtained by simple texture coordinate additions. For a circular jittered kernel, however, aligning the samples one by one can be expensive and cancel the jitter effect. In this case, we recommend calculating the virtual samples at the exact coordinates given by the PCF kernel, without alignment to the texel centers. This is usually enough to prevent self-shadowing artifacts with a small constant bias.

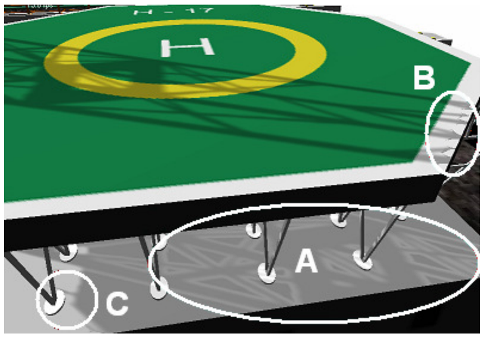


Figure 2. Light Bleeding with VSMs

2.3. Warping or parameterization

Several techniques attempt to minimize aliasing artifacts by warping the shadow map to an appropriate space before sampling the scene. In this work, we focus on perspective parameterization techniques, introduced by Stamminger & Drettakis [13]. The idea is to apply a perspective transformation to the scene in such a way that objects closer to the viewer (not necessarily close to the light source) appear in a larger area of the shadow map. The techniques have only a very low cost on CPU to compute the transform matrix and can drastically reduce shadow border aliasing in most cases.

In order to understand perspective warping and its benefit, we must first define an aliasing error metric. Consider Figure 3, where a surface area can be seen by both the camera and the light source. The camera and light beams pass through one image pixel and one shadow map texel, respectively, and hit the surface with angles θ_c and θ_L, covering areas w_c' and w_L'.

Note that, when $w_L' > w_c'$, a single shadow map texel determines the shadow state of more than one image pixel. Therefore, we can define the aliasing error m as the ratio between the areas "seen" by a shadow map texel and an image pixel [5]:

$$m = \frac{w_L'}{w_c'} \approx \frac{w_L \cos(\theta_c)}{w_c \cos(\theta_L)} \tag{3}$$

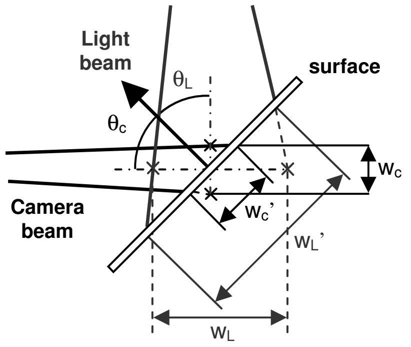


Figure 3. Aliasing error metric

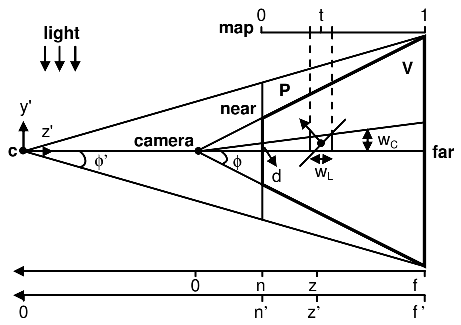
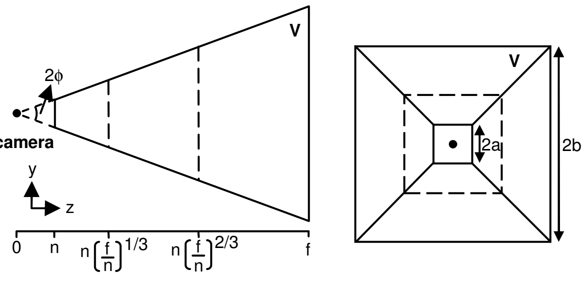


Figure 4. Perspective warp P fit to a frustum V


V
2ϕ
camera
y
z
0
n
$n\left[\frac{f}{n}\right]^{1/3}$
$n\left[\frac{f}{n}\right]^{2/3}$
f
V
2a
2b

(a) perpendicular light (b) parallel light

Figure 5. View frustum, as seen by the light source, with z-partitioning

Equation (3) is often used to classify aliasing into *perspective aliasing* (w_L/w_c) and *projection aliasing* ($\cos(\theta_L)/\cos(\theta_c)$) [13]. The second type is potentially unbounded and depends on the orientation of each surface of the scene in relation to the light and view directions. Since this effect is also less noticeable and not avoided by perspective warping, it is usually ignored in the analysis of such algorithms.

Now, consider Figure 4, where V is the view frustum. The light and view directions are perpendicular in this case. Perspective warping techniques first fit a frustum P around the view volume (and objects that cast shadows into it). Before updating the shadow map, they apply the perspective transformation associated with P to the scene. In the figure, V has a field-of-view of 2ϕ, near plane n and far plane f. Similarly, P has a field-of-view of $2\phi'$, near plane n' and far plane f'. A generic point on the central axis of V, which is at a distance z from the camera, is mapped to a distance z' from point c, the center of projection of P. The coordinates (x',y',z') correspond to the "light space" defined in the LiSPSM paper [2], whose origin gives the position of c. The shadow map coordinates (s,t) are assumed to be aligned with (x',z'), respectively.

The strength of the perspective warp is controlled by the parameter n'. When $n' = n$, we have the PSM (Perspective Shadow Map) technique [13]. When $n' \rightarrow \infty$, the warp loses strength and becomes orthogonal, leading to the standard shadow mapping. The LiSPSM (Light-Space Perspective Shadow Map) parameter $n'_{LiSPSM} = n + \sqrt{f\,n}$ is the one that minimizes M_z, the maximum error in the z direction:

$$M_z = \max_{z \in [n,f]} \left(w_{Lz} / w_{cy} \right) \qquad (4)$$

It can be shown [2,5] that, while PSM minimizes the maximum error in the x direction, it makes the error in the z direction grow linearly with z, which produces strong aliasing on surfaces not close to the near plane. On the other hand, LiSPSM minimizes the maximum error in the z direction while also keeping a small maximum error in the x direction. Furthermore, LiSPSM guarantees that the error in the z direction is equal on both the camera's near and far planes, which is beneficial for z-partitioning (Section 2.4). For those reasons, we recommend the use of LiSPSM for CAD model rendering and other applications.

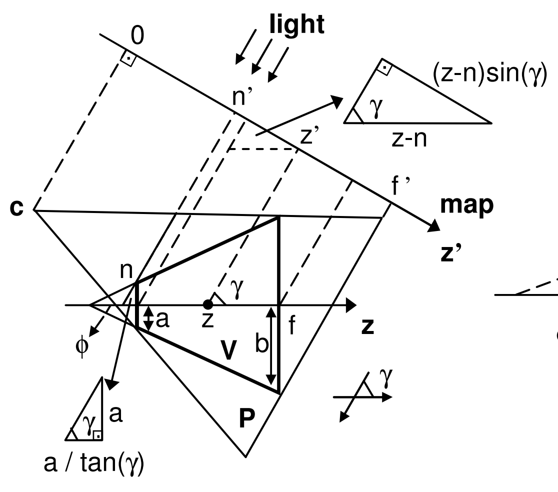
2.4. Partitioning

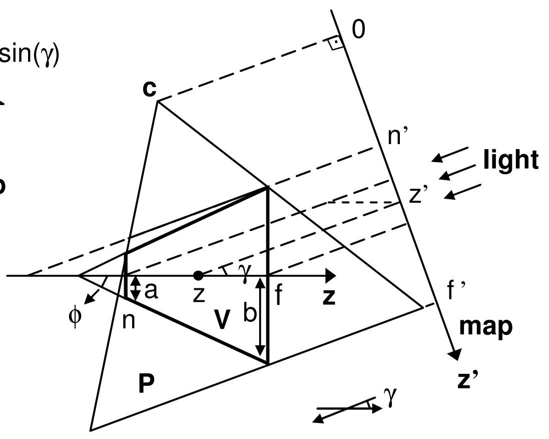
Partitioning techniques subdivide the scene according to some criteria and generate a different shadow map for each partition. The shadow map resolution allocated to each partition may be the same or be decided by an aliasing error metric.

In this section, we focus on face and z-partitioning as described by Lloyd et al. [4,5]. Compared with other techniques [14,15], these are simpler to implement, require fewer rendering passes, and can more easily be combined with warping for better results.

Every perspective warping technique fails when the light and view directions are near parallel. In this situation, the parameterization is usually forced to converge to the standard shadow mapping by imposing $n' \rightarrow \infty$ [2,5]. This difficulty can be overcome by dividing the view frustum according to its faces, as seen by the light source, and generating a different shadow map for each division with its own transformation [4,5]. This "face partitioning" approach, however, can be too expensive, as it requires at least four shadow maps to be generated each frame. Besides, despite its excellent results in the near-parallel case, the technique does not improve shadow quality when the light is perpendicular to the view direction.

As a different approach, z-partitioning attempts to achieve higher shadow quality in all cases by dividing the view frustum along its z axis, as shown in Figure 5. A shadow map, either warped or not, is then generated for each partition. This can be seen as a piecewise approximation of the optimal logarithmic parameterization of the shadow map [5], which gives constant aliasing error in both x and z directions.





(a) $\phi \le \gamma \le \pi/2$ (b) $0 < \gamma < \phi$

Figure 6. Shadow Map generalized perspective parameterization

With z-partitioning, the maximum aliasing error in each partition is proportional to its f / n ratio. As a result, the optimal z-partitioning scheme uses self-similar partitions, that is, which have the same f / n ratio [4,5], as in Figure 5a. For a total of k partitions, the i'th partition satisfies:

$$n_i = n\left(\frac{f}{n}\right)^{(i-1)/k}, f_i = n_{i+1} = n\left(\frac{f}{n}\right)^{i/k}, i \in \{1,2,...,k\} \quad (5)$$

As shown by Lloyd et al. [5], z-partitioning can provide more benefit than face-partitioning with the same number of subdivisions. Moreover, the use of only 2 or 3 partitions with LiSPSM warping suffices to greatly reduce aliasing errors when the light is either perpendicular or parallel to the view direction. This is thus the recommended setting for most systems.

3. Proposed improvements

In this section, we propose two improvements to existing techniques. First, we build on [2,3,5] to derive a LiSPSM parameter generalized to different angles between the light and view directions. Second, we propose an adaptive z-partitioning scheme based on the work of Lloyd et al. [4,5].

3.1. Generalized LiSPSM parameter

Recall that Figure 4 shows the optimal configuration for the LiSPSM warp, when the light and view directions are perpendicular. Based on the figure and on our aliasing error metric (Equation 3), it can be shown [5] that, for an image of *($res_{cx} \times res_{cy}$)* pixels and a shadow map of *($res_{ls} \times res_{lt}$)* texels, the maximum error in the *z* direction is given by:

$$m_z = \frac{w_{lz}}{w_{cy}} = \frac{res_{cy}}{res_{lt}} \frac{1}{2\tan(\phi)} \frac{1}{z}\left(\frac{dt}{dz}\right)^{-1} \quad (6)$$

The derivative in Equation (6) is calculated from the standard OpenGL perspective transformation:

$$t(z') = \frac{1}{2} + \frac{1}{2}\frac{(f'+n')}{(f'-n')} - \frac{f'n'}{z'(f'-n')} \quad (7)$$

Figure 4 also gives:

$$z' = n' + z - n \quad (8)$$
$$f' = n' + f - n \quad (9)$$

Equations (8) and (9) are only valid for the situation depicted in Figure 4, where the light and view directions are perpendicular. If the light is tilted towards or away from the camera, we must consider the two general cases shown in Figure 6, where the smallest angle between the light and view directions is γ. Aided by the auxiliary triangles shown in Figure 6a, we can see that:

$$z' = \begin{cases} n' + a\cos(\gamma) + (z-n)\sin(\gamma), & \phi_y \le \gamma \le \pi/2 \\ \mathrm{n}' + b\cos(\gamma) + (z-f)\sin(\gamma), & 0 < \gamma < \phi_y \end{cases} \quad (10)$$

$$f' = \begin{cases} n' + (a+b)\cos(\gamma) + (f-n)\sin(\gamma), & \phi_y \le \gamma \le \pi/2 \\ n' + 2b\cos(\gamma) & 0 < \gamma < \phi_y \end{cases} \quad (11)$$

In these equations, a and b are half the height of the view frustum's near and far planes, respectively. Substituting (10) into (7), differentiating, and using the result in (6), we find the generalized error in the z direction:

$$m_z(z,n',\gamma) = \frac{w_{lz}}{w_{cy}} = \xi(n',\gamma)\frac{(\psi(z,n',\gamma))^2}{z} \qquad (12)$$

$$\xi(n',\gamma) = \frac{res_{cy}}{res_{lt}}\frac{1}{2\tan(\phi_y)}\frac{(f'-n')}{n'f'}\frac{1}{\sin(\gamma)}$$

$$\psi(z,n',\gamma) = \begin{cases} n'+a\cos(\gamma)+(z-n)\sin(\gamma), & \phi_y \le \gamma \le \pi/2 \\ n'+b\cos(\gamma)+(z-f)\sin(\gamma), & 0<\gamma<\phi_y \end{cases}$$

Differentiation of Equation (12) shows that the error function has only one local minimum in $z \in [n,f]$. The maximum error must then occur at the camera's near and far planes, and can be minimized by making its value equal on those extremes:

$$m_z(n,n',\gamma) = m_z(f,n',\gamma)$$
$$\Rightarrow n' = \begin{cases} n_1'(\gamma) = (n+\sqrt{fn})\sin(\gamma) - a\cos(\gamma), & \phi_y \le \gamma \le \pi/2 \\ n_2'(\gamma) = (f+\sqrt{fn})\sin(\gamma) - b\cos(\gamma), & 0<\gamma<\phi_y \end{cases} \qquad (13)$$

Equation (13) represents a generalized LiSPSM parameter that keeps the maximum error minimized when $\gamma < \pi/2$. Unfortunately, however, n_2' assumes negative values when the light and view directions are near parallel. This means that our constraint is impossible to satisfy, which corroborates the fact that every perspective warp must converge to the standard shadow mapping in this case [2,3,5]. In order to obtain a smooth transition in the shadow quality, we modify Equation (13) using the following new constraints:

$$\lim_{\gamma\to 0} n_2' = \infty$$
$$n_2'(\gamma_{\lim}) = n_1'(\gamma_{\lim}) \qquad (14)$$

In Equation (14), γ_{lim} is the limit angle where we stop using Equation (13). To avoid numerical instability and the generation of overly wide warp frustums, it should be kept higher than ϕ. We recommend using the angle which makes the near and far faces of the view frustum superpose in the light's view:

$$\gamma_{\lim} = \arctan\left(\frac{a+b}{f-n}\right) \qquad (15)$$

Our modified parameter can now be written as:

$$n' = \begin{cases} n_1'(\gamma) = (n+\sqrt{fn})\sin(\gamma) - a\cos(\gamma), & \gamma_{\lim} \le \gamma \le \pi/2 \\ n_2'(\gamma) = \dfrac{n_1'(\gamma_{\lim})}{r(\gamma)}, & 0<\gamma<\gamma_{\lim} \end{cases} \qquad (16)$$

In Equation (16), the function $r(\gamma)$ descends from $r(\gamma_{lim}) = 1$ to $r(0) = 0$, which satisfies the constraints of Equation (14). In their paper, Zhang et al. propose the following function for the PSM case [3]:

$$r(\gamma) = \sin\left(\frac{\gamma}{\gamma_{\lim}}\frac{\pi}{2}\right) \qquad (17)$$

For our LiSPSM case, however, we required $r(\gamma)$ to descend faster to zero, once again to avoid excessively wide warping frustums. We have achieved good results with the following function:

$$r(\gamma) = \left[1+\sin\left(\left(\frac{\gamma}{\gamma_{\lim}}-1\right)\frac{\pi}{2}\right)\right]^2 \qquad (18)$$

Figure 7 shows the form of the final generalized LiSPSM parameter. The warp at first becomes stronger as the light is tilted towards or away from the view direction. Below the limit angle γ_{lim}, the warp quickly converges to the standard shadow mapping.

In general, we have achieved better shadow quality with our generalized LiSPSM than with other parameterizations for most light directions. Figure 8 compares the standard and generalized LiSPSM warps for a lattice shadow close to the near plane, where the improvement is most noticeable. Nevertheless, one must keep in mind that minimizing the maximum error over the entire view frustum does not guarantee a minimal error everywhere in the scene. A simpler technique like the original LiSPSM or PSM might give better results for specific cases.

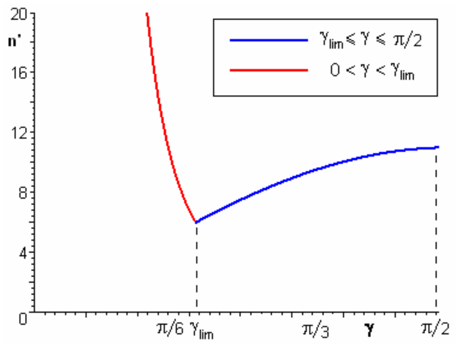


Figure 7. Generalized LiSPSM parameter *(n = 1, f = 100, $\phi_y = \pi/6$).*

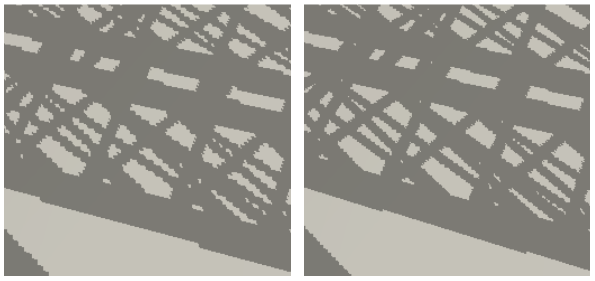


Figure 8. Left: Standard LiSPSM. Right: Generalized LiSPSM.

3.2. Adaptive z-partitioning

Recall the z-partitioning scheme described in Section 2.4. When the light and view directions are perpendicular, the use of self-similar partitions and of LiSPSM warping guarantees that the maximum aliasing error in the *z* direction is minimal and equal in the near and far planes of every partition. In the *x* direction, the error is small and does not change drastically over the view frustum. These are important properties because otherwise a sudden change in the shadow quality would be noticeable as a "seam" at the interface between partitions.

However, when the light and view directions are not perpendicular, the LiSPSM warp converges to the standard shadow mapping. As a result, the aliasing error in both directions has its maximum value at the near plane and minimum value at the far plane of each partition. In this case, the algorithm produces visible seams between each pair of partitions, as shown by the detail on the left in Figure 9. Moreover, the seams move together with the camera, which catches the user's attention and becomes very distracting.

In order to minimize seams, the ratio f / n of all partitions, except the one closest to the camera, should be kept small. This reduces the difference between the maximum error at the near plane and the minimum error at the far plane. The disadvantage is that the error close to the camera (near plane of the first partition) becomes higher than with the optimal partitioning.

In this paper, we propose an adaptive partitioning scheme. When the light and view directions are perpendicular, the optimal z-partitioning should be used to minimize the maximum error over the view frustum. In the near-parallel case, however, every *i*'th partition but the first should have its (f_i / n_i) ratio reduced to a small value α_i. A smooth transition can be achieved by linearly interpolating the distance of each subdivision between the optimal and limit cases, using the angle γ between the light and view directions as the weight:

$$\lambda = \frac{\gamma}{\pi/2}, \qquad 0 \le \lambda \le 1$$

$$\begin{cases} f_k = f \\ n_i = f_{i-1} = \lambda \left[n \left(\dfrac{f_i}{n} \right)^{\frac{i-1}{i}} \right] + (1-\lambda)\dfrac{f_i}{\alpha_i}, \quad i \in \{k,\ldots,3,2\} \\ n_1 = n \end{cases} \tag{19}$$

Note that Equation (19) is calculated iteratively, starting with the last partition *(i=k)* and backward until the second *(i=2)*, always using the values of f_i previously calculated by the linear interpolation. The first partition has $n_1=n$ and $f_1=n_2$. It can be shown that, when $\gamma = \pi/2$, Equations (5) and (19) are equivalent.

Figure 9 compares the shadows generated by the optimal z-partitioning technique and our adaptive one with 2 partitions. The interface between the partitions is indicated with a white arrow. In the left detail, optimal partitioning produces the best shadow quality close to the far plane of the first partition. However, the technique produces a visible seam between this region and the beginning of the next partition. When using the proposed adaptive scheme with $\alpha=2$, aliasing is not reduced so dramatically in the first partition, but the seam becomes almost unnoticeable.

Empirically, we recommend a value α no higher than 4 without PCF filtering and no higher than 2 with PCF, since the filter in texel space as described in Section 2.2 makes seams more noticeable (note the blur in the second partition in the left of figure 9).

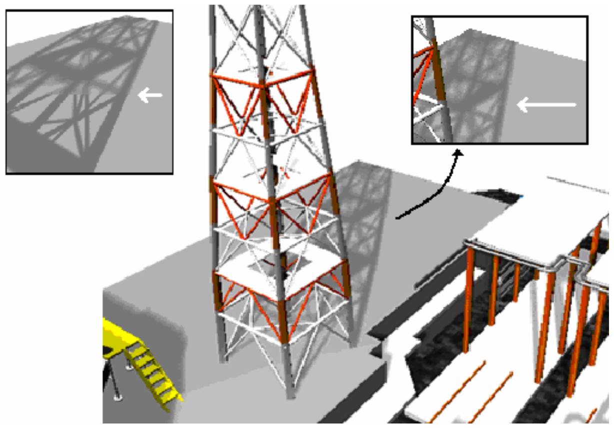


Figure 9. Adaptive z-partitioning with detail on the right. The detail on the left shows the result of optimal z-partitioning with visible seam

4. Results and additional remarks

We have implemented all the techniques discussed thoroughly in this paper on a machine with a dual core 2.4GHz processor and a GeForce 8800 GTS graphics card. Tests were performed on an oil platform CAD model with 250k triangles, using 800×600 screen resolution and 2048×2048 shadow maps.

For our performance tests, we used constant bias, generalized LiSPSM warping, 2 adaptive partitions along the view frustum's *z* axis and 5×5 texel-space PCF with jittered kernel samples and multiple non-aligned virtual sample generation. Despite the extra cost associated with the generation of two shadow maps and the use of PCF in the final rendering pass, these techniques can run faster than the standard shadow mapping for some camera and light views, as shown in Table 2, since frustum culling can be optimized for each partition. The results in the table were obtained for a close-up view where about 20% of the scene's geometry was visible to the camera, while the light sources circled around the scene at a far distance. From our results, it should also be noted that, due to the increasing fragment processing power of graphics cards, PCF can be applied with nearly negligible cost.

As a final remark, it is important to notice that, when aliasing is not completely eliminated, view-dependent warping techniques can cause shadow borders to crawl while the camera or light moves around. Fortunately, blurring the shadow borders with PCF makes this effect much less noticeable. However, the situation of near-parallel light and view directions must still be handled with care to avoid disturbing temporal changes in shadow quality. This emphasizes the importance of a good partitioning scheme.

Table 2. Performance results

Technique / Lights	Standard Shadow Map	Recommended Techniques
1 dynamic light	45 fps	29 to 59 fps
2 dynamic lights	25 fps	18 to 30 fps

5. Conclusion

We have presented a survey of shadow mapping techniques that attempt to reduce self-shadowing and aliasing artifacts. In particular, we have investigated the effectiveness of depth bias, sample alignment, filtering, perspective warping and partitioning techniques when applied to CAD model visualization. For this kind of application, we recommend the use of a constant bias or the dual-depth technique when generating the shadow map, as well as LiSPSM warping and 2 or 3 partitions along the view frustum's z axis. In the final rendering pass, we recommend the use of texel-space PCF with a 3×3 to 5×5 jittered kernel and multiple non-aligned virtual samples. We have concluded that this set of techniques can produce high quality shadows in real-time for CAD models.

Additionally, we have presented improvements to existing techniques: a generalized parameter for LiSPSMs, which keeps the maximum error in the z direction minimized for different angles between the light and view directions; and an adaptive z-partitioning scheme that minimizes seams, that is, avoids sudden changes in shadow quality between each pair of partitions.

Acknowledgements

During this research, the first author was financially supported by Brazilian agency CNPq. Tecgraf is a laboratory at PUC-Rio that is mainly funded by the Brazilian oil company, Petrobras.

References

[1] L. Williams, "Casting curved shadows on curved surfaces", *Proceedings of ACM SIGGRAPH'78*, ACM Press, 1978, vol. 12, pp. 270-274.

[2] M. Wimmer, D. Scherzer, and W. Purgathofer, "Light space perspective shadow maps", *Proceedings of the Eurographics Symposium on Rendering 2004*, Eurographics Association, 2004, pp. 143-152.

[3] F. Zhang et al., "Generalized Linear Perspective Shadow Map Reparameterization", Proceedings of the ACM international conference on virtual reality continuum and its applications, 2006. pp. 339-342.

[4] B. Lloyd et al., "Subdivided Shadow Maps", University of North Carolina at Chapel Hill, 2005, Technical Report TR05-024.

[5] B. Lloyd et al., "Warping and Partitioning for Low Error Shadow Maps", Proceedings of the Eurographics Symposium on Rendering 2006, Eurographics Association, 2006, pp. 215-226.

[6] Y. Wang, and S. Molnar, "Second-Depth Shadow Mapping", The University of North Carolina at Chapel Hill, 1994, Technical Report TR94-019.

[7] A. Woo, "The Shadow Map Revisited". D. Kirk (ed.), Graphics Gems III, AP Professional, Boston, 1992, pp. 338-342.

[8] D. Weiskopf, and T. Ertl, "Shadow Mapping Based on Dual Depth Layers", Proceedings of the Eurographics 2003 Short Papers, Eurographics Association, 2003, pp. 53-60.

[9] W. Reeves, D. Salesin, and R. Cook, "Rendering antialiased shadows with depth maps", Proceedings of ACM SIGGRAPH'87, ACM Press, 1987, vol. 21, pp. 283-291.

[10] W. Donnelly, and A. Lauritzen, "Variance shadow maps", Proceedings of the 2006 symposium on Interactive 3D graphics and games, ACM Press, New York, NY, USA, 2006, pp. 161-165.

[11] http://forum.beyond3d.com/showthread.php?t=38165, Internet forum, last access on 07/20/2007.

[12] Y. Kryachko, "Efficient Soft-Edged Shadows Using Pixel Shader Branching", M. Phar, and R. Fernando (Ed.), GPU Gems II, ch. 17, NVidia Corp., 2005, pp. 269-282.

[13] M. Stamminger, and G. Drettakis, "Perspective shadow maps", Proceedings of ACM SIGGRAPH'02, ACM Press, 2002, pp. 557-562.

[14] J. Arvo, "Tiled shadow maps", Proceedings of Computer Graphics International 2004, IEEE Computer Society, 2004, pp. 240-247.

[15] R. Fernando et al., "Adaptive shadow maps", Proceedings of ACM SIGGRAPH, ACM Press, 2001, pp. 387-390.