Subdivided Shadow Maps

Technical Report TR05-024

Brandon Lloyd, Sung-eui Yoon, David Tuft, Dinesh Manocha
University of North Carolina at Chapel Hill

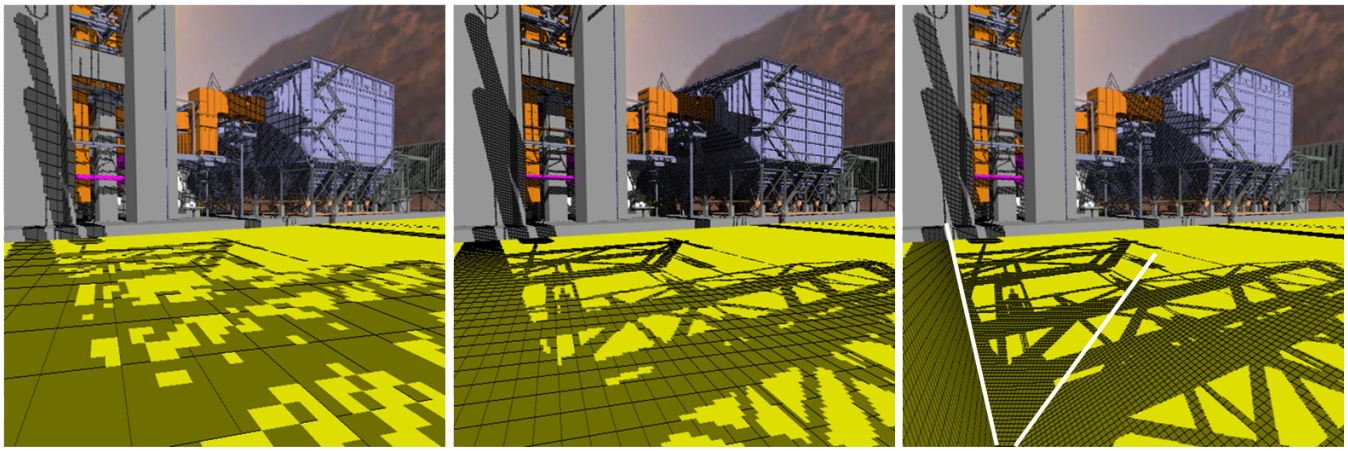

Figure 1: Comparison between trapezoidal shadow maps (TSMs) and subdivided shadow maps. a) TSM with 1K×1K shadow map, b) TSM with 4K×4K shadow map, and c) 1K×1K subdivided shadow map. This configuration with a small angle between the light and view directions is difficult for prior methods. Even with the largest shadow map that can be allocated on current hardware, TSMs are not able to match the quality of subdivided shadow maps for this view.

Abstract

We present a technique for reducing perspective aliasing error in shadow maps. From the viewpoint of the light, the scene is first split into subdivisions defined by the visible faces of the camera frustum. The frustum subdivisions may be further subdivided along their corresponding faces. We apply a separate shadow map warp to each resulting subdivision. This produces significantly less error than applying a single shadow map warp to the whole scene We layout the subdivisions in rectangular regions within a single shadow map, using the maximum error of each subdivision to assign larger regions to subdivisions with higher error. Our method runs well on commodity graphics hardware and is easy to integrate into existing shadow map systems. We are able to achieve interactive performance (8-25 fps) on a power plant model (12M triangles), a double eagle tanker model (82M triangles), and the St. Matthew model (370M triangles) running on a PC with a GeForce 7800 GTX. We observed significantly less aliasing compared to prior shadow map warping algorithms.

1 Introduction

Shadows are an important component of an interactive rendering system. They add realism and important visual cues. In this paper we restrict ourselves to hard shadows generated by directional light sources using shadow maps. For the directional light source, the standard shadow map algorithm proposed by Williams [1978] renders the scene with an orthographic projection from the light's view, and uses the resulting depth map to determine which surfaces lie in shadow. Shadow maps are a particularly attractive algorithm because they are easy to implement, they support a wide variety of geometry representations, and there exists wide support for shadow maps in current graphics hardware. The main drawback of shadow maps is aliasing at the shadow edges. Shadow map aliasing caused by the orientation of the surface onto which the shadow is projected is called *projective aliasing*. Aliasing due to limited shadow map resolution is called *perspective aliasing*. A number of algorithms have been proposed that address perspective aliasing by warping the shadow map so as to allocate more shadow map resolution near the viewer where the aliasing is the worst [Stamminger and Drettakis 2002; Wimmer et al. 2004; Martin and Tan 2004]. The warping is performed by applying a transformation to the scene before it is rendered into the shadow map. The warping reduces aliasing artifacts for most light/camera configurations. However when the angle between the light and view directions is small, all previous warping algorithms revert back to standard shadow maps, resulting in large perspective aliasing error.

Main Results: In this paper we present an algorithm to reduce perspective aliasing error by subdividing the scene and applying a separate warping function to each subdivision. Our algorithm uses two types of subdivisions:

- **Frustum subdivisions.** From the light's view, the algorithm first subdivides the eye view frustum into subdivisions corresponding to the visible frustum faces.
- **Face subdivisions.** The frustum subdivisions may be further subdivided along the length of their corresponding faces.

Frustum subdivisions dramatically reduce the error for configurations with small angles between the light and view

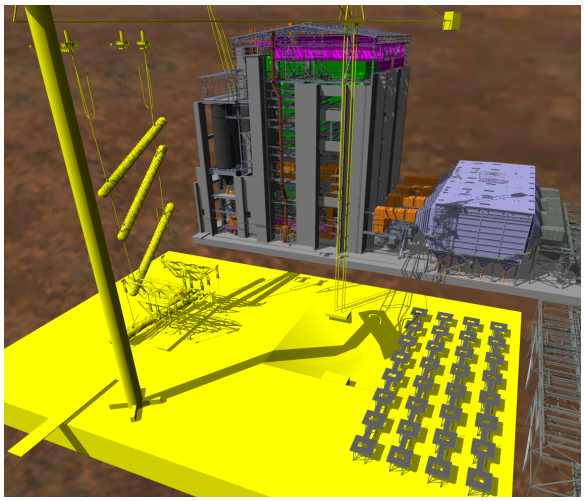

Figure 2: **Power plant model.** *This image shows a power plant model consisting of more than* 12 *million triangles. We are able to render the model with high quality shadows at* 8-20 *frames per second on a PC with GeForce 7800 GTX.*

directions. Face subdivisions reduce error for all configurations. By warping face subdivisions we more closely approximate the optimal warping function that would completely eliminate aliasing.

We formulate the maximum error in each subdivision. Using this information we can allocate to each subdivision a rectangular region in the shadow map. The regions are proportional in size to the maximum error of their corresponding subdivisions. Allocating space in the texture map in this way insures that the maximum error over the entire scene in minimized. Once the subdivisions have been rendered into their separate regions of the shadow map, a fragment program is used to render the final image.

Compared to previous shadow map warping algorithms, subdivided shadow maps provide a more even distribution of error over all light directions and lower error overall, with only a modest increase in rendering cost. We have integrated our algorithm with a rendering system for large models that runs on commodity hardware. With a GeForce 7800 GPU our system achieves interactive performance (8-25 fps) on a power plant model consisting of 12 million triangles, a double eagle tanker model consisting of 82 million triangles, and the St. Matthew model consisting of 372 million triangles. We found that the performance with the subdivided shadow map was only about twice as slow as with a standard shadow map. We were able to reduce the maximum aliasing error by a factor of up to 10 times.

The rest of this paper is organized as follows. In Section 2 we briefly discuss related work in shadow map computation. Section 3 provides the background for quantifying the perspective aliasing error and the parameterization of the warping functions that we use. The subdivision algorithm is described in Section 4. We discuss various implementation details and our results in Section 5. Finally, we conclude with some ideas for future work.

2 Previous Work

Many techniques have been proposed for shadow generation. In this section, we limit ourselves to shadow maps and some hybrid combinations with object-space techniques. Shadow maps were first introduced by Williams [1978]. Segal et al. [1992] later implemented them on standard graphics hardware. Many algorithms have been proposed to address the aliasing problems with shadow maps. Adaptive shadow maps [Fernando et al. 2001] use a hierarchy of small shadow maps to allocate resolution where it is required. Increased programmability of GPUs has facilitated implementations of adaptive shadow maps for hardware rendering [Lefohn et al. 2005], but performance can be slow. Instead of using a regular grid of shadow samples, irregular shadow maps [Aila and Laine 2004; Johnson et al. 2004] use a sample distribution that corresponds exactly to the image samples for the eye, thus avoid the aliasing problem altogether. However, irregular shadow maps are difficult to implement on current hardware. Shadow map warping was introduced with perspective shadow maps (PSMs) [Stamminger and Drettakis 2002]. PSMs use the camera's perspective transform to warp the shadow map. A singularity may arise with PSMs that requires special handling [Kozlov 2004]. Light-space perspective shadow maps (LSPSMs) [Wimmer et al. 2004] are a generalization PSMs that do not have the singularity problem because they use a perspective projection that is oriented perpendicular to the light direction. Trapezoidal shadow maps (TSMs) [Martin and Tan 2004] are very similar to LSPSMs, except that they use a different formulation for the perspective projection. Chong et al. [2004] use a general projective transform to ensure that there is a one-to-one correspondence between pixels in the image and the texels in the shadow map, but only for a small number of planes within the scene. Others have discussed using separate shadow maps for different parts of the scene [Tadamura et al. 1999; Forum 2003; Aldridge 2004].

Pure object-space shadow algorithms do not have aliasing problems. Some hybrid algorithms combine object-space techniques with shadow maps to reduce aliasing. McCool et al. [2000] construct shadow volumes from a shadow map. Sen et al. [2003] create a shadow map that more accurately represents shadow edges. Both of these techniques, while generating better looking shadow edges, may miss small features that cannot be represented in the shadow map. Chan and Durand [2004] use shadow maps to restrict shadow volume rendering to the shadow edges. Govindaraju et al. [2003] use shadow polygons for the most aliased areas and a shadow map everywhere else.

3 Shadow map warping

In this section we review perspective aliasing, we show how aliasing can be controlled by reparameterizing the shadow map, and we quantify the error. We also present our own parameterization of the warping function and derive the error equations. We follow the general outline of the excellent analysis provided by Wimmer et al. [2004], but our explanation is cast in somewhat different terms. Our notation also differs slightly.

3.1 Shadow map warping

We illustrate the concept of shadow map warping using the 2D configuration shown in Figure 3. A directional light source is positioned perpendicular to the view direction of a

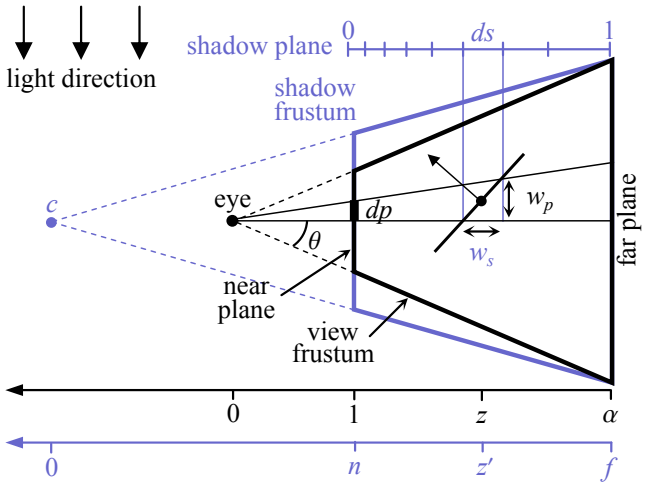

Figure 3: **Perspective aliasing and the shadow map warping parameterization.** *Perspective aliasing occurs when shadow texel beams are wider than pixel beams. The shadow map is warped using a perspective projection placed around the view frustum. This is a canonical view frustum with near plane at* 1*. The parameter* α *represents the ratio of the far to near plane distances.*

perspective camera. The shape of the view frustum is parameterized by α, the near to far plane distance ratio. This will permit us later to more easily understand how the aliasing errors change when we change the shape of the frustum.

Perspective aliasing. The notion of perspective aliasing with a shadow map can be understood intuitively in terms of the relative widths of shadow and pixel beams. In Figure 3 parallel shadow beams emanate from the texels of the shadow plane with widths w_s. Pixel beams likewise emanate from the pixels on the view plane. The widths of the pixel beams w_p increase with z. If shadow and pixel beams fall on a surface that is oriented with its normal half-way between the view and light directions, the size of the resulting footprints depends only on the relative widths of the beams, $m = w_s/w_p$. There is no projective aliasing in this case. When the shadow beams are wider than the pixel beams, i.e. $m > 1$, a shadow beam footprint is covered by multiple pixels beams. Thus the footprints of individual shadow texels can be clearly distinguished in the image and perspective aliasing occurs. Ideally we would like $w_s = w_p$ everywhere in the scene in order to avoid aliasing and to make the best use of the shadow map resolution.

Parameterization. As shown in Figure 3, we would like to reparameterize the shadow plane so that the widths of the shadow texel beams more closely match those of the pixel beams they intersect. Since texel spacing is inversely proportional to the derivative of the texture parameterization, the width of a shadow texel beam can be expressed as:

$$w_s = \frac{1}{r_s}\frac{dz}{ds},$$

where r_s is the resolution of the shadow map. The width of a pixel beam depends on the image pixel resolution r_p and

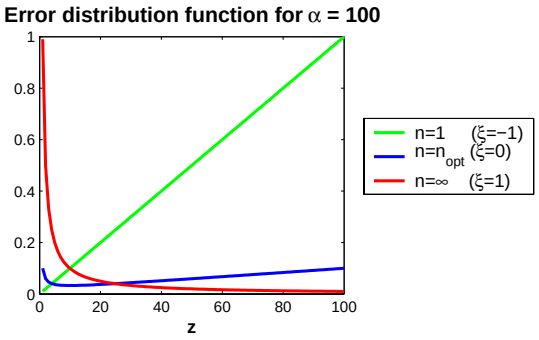

Figure 4: **Error distribution function.** *This graph shows* g *plotted for various values of* n *and corresponding* ξ*.*

is proportional to z.

$$w_p = z\frac{2\tan\theta}{r_p}.$$

To minimize the perspective aliasing error, m, we would like to have the shadow texel width also be proportional to z. In other words, we seek a parameterization of the shadow plane s whose derivative ds is proportional to $1/z$:

$$s = \int_0^1 ds = \int_1^z \frac{1}{z}dz = \ln z.$$

Thus the optimal parameterization of the shadow plane is a logarithmic function. The only non-linear transform that current graphics hardware provides is a perspective projection, which gives texel spacing proportional to $1/z^2$ (see Appendix). Therefore, the best we can do is to minimize the resolution mismatch error according to some optimality criterion.

The parameterization of the shadow plane with a perspective projection can be specified in a manner similar to that used for the camera. With near and far planes that coincide with those of the camera, the parameterization is given by:

$$s = \frac{1}{2} + \frac{f+n}{2(f-n)} + \frac{fn}{z'(f-n)}.$$

The position of the center of projection c is controlled by near plane distance n, which is a free parameter. When $n = 1$, c coincides with the eye and the warping effect is strongest. At $n = \infty$ the perspective projection becomes orthogonal and the texel spacing becomes uniform as in the standard shadow mapping algorithm. The fundamental difference between the recently proposed shadow map warping algorithms (PSM[Stamminger and Drettakis 2002], LSPSM[Wimmer et al. 2004], and TSM[Martin and Tan 2004]) is the choice of n. PSMs place n at 1. LSPSMs choose n such that the maximum error for the whole frustum is minimized. TSMs choose n such that a user specified portion at the front of the frustum is mapped to the 80% line on the shadow plane.

Quantifying error. The choice of the n parameter greatly affects the perspective aliasing error m. To compute m we first replace z' in the parameterization function s with $(z - 1 + n)$ and differentiate:

$$\frac{ds}{dz} = \frac{fn}{(z-1+n)^2(f-n)}. \quad (1)$$

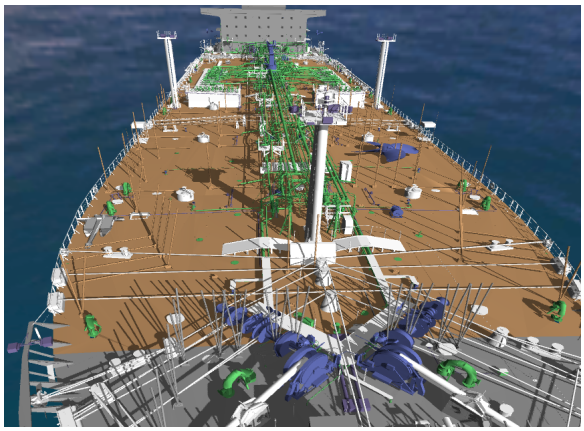

Figure 5: **Double eagle tanker model.** *This image shows a double eagle tanker model consisting of* 82 *million triangles. We are able to render the model at* 5-10 *frame per second with high shadow quality.*

If we substitute $f = n + \alpha - 1$ into Equation 1 we obtain for m:

$$m = \frac{w_s}{w_p} = \frac{r_p}{r_s}\,\frac{(\alpha-1)}{2\tan\theta}\,g \tag{2}$$

$$g = \frac{(z-1+n)^2}{zn(n+\alpha-1)} \tag{3}$$

The first term captures the effect of the relative resolutions of the image and the shadow map. The second term is characterized by the shape of the camera frustum. The last term, g is the error distribution function. Figure 3.1 shows g plotted over the range of the frustum $z \in [1, \alpha]$ for various values of n. The function g reaches its maximum value of 1 on the far and near planes with $n = 1$ and $n = \infty$ respectively. The least maximum value of g, $g_0 = \min_n \max_z(g) = 1/\sqrt{\alpha}$, occurs at the optimal n parameter for LSPSMs $n_{opt} = 1 + \sqrt{\alpha}$.

3.2 Reparameterizing the error distribution function

We reparameterize the error distribution function g to facilitate the error analysis for subdivision and to provide a more intuitive control over the maximum error. The n parameter is cumbersome because it is tied to scale of the view frustum and it is not directly related to the maximum error. We observe that as n decreases from infinity to 1, the maximum value of g over the whole frustum $\max_z(g)$ moves from 1 on the near plane to g_0 and back up to 1 again on the far plane. Based on this behavior we choose a new parameter ξ that controls the value and location of the maximum error. Over the range $\xi \in [-1, 0]$, we want $g_{\max}$ to linearly interpolate on the near plane from the maximum value of 1 to the minimum g_0 . For $\xi \in [0, 1]$, $g_{\max}$ should move back to 1 on the far plane.

$$\xi = \begin{cases} \frac{g_0 - g(n,1)}{1-g_0} = \frac{\sqrt{\alpha}-n+1}{n+\alpha-1}, & n \geq n_{opt}, \\ \frac{g(n,\alpha)-g_0}{1-g_0} = \frac{\sqrt{\alpha}-n+1}{n\sqrt{\alpha}}, & n < n_{opt}. \end{cases}$$

We can then solve for n in terms of ξ:

$$n = \begin{cases} \frac{\sqrt{\alpha}+1-\xi(\alpha-1)}{\xi+1}, & \xi \leq 0, \\ \frac{\sqrt{\alpha}+1}{\xi\sqrt{\alpha}+1}, & \xi > 0. \end{cases} \tag{4}$$

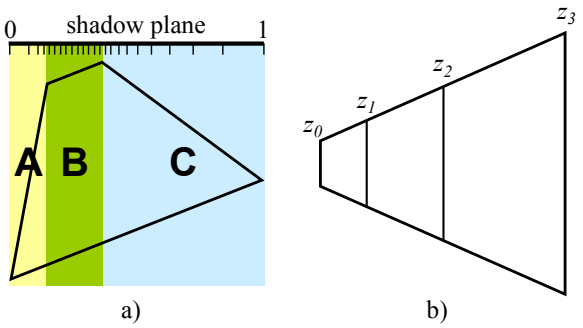

Figure 6: **Subdivisions.** ***a)*** Frustum subdivisions *divide the frustum into faces over which the texel spacing increases monotonically.* ***b)*** Face subdivision *split along a face to more closely approximate the optimal shadow map parameterization.*

Plugging these values for n back into Equations 2 and 3 with $z = \alpha$ for $\xi \leq 0$ and $z = 1$ for $\xi > 0$, we get an expression for the maximum error over the entire frustum in terms of ξ:

$$m_{\max} = \frac{r_p}{r_s}\,\frac{(\alpha-1)}{2\tan\theta}\,\frac{1+|\xi|(\sqrt{\alpha}-1)}{\sqrt{\alpha}}. \tag{5}$$

4 Subdivision Algorithm

Subdividing the scene and applying a separate warping function to each subdivision can drastically reduce perspective aliasing error. In this section we discuss two types of subdivision: frustum subdivision and face subdivision.

Frustum subdivision. Till now we have only considered the configuration with the light direction perpendicular to the view direction. Figure 6(a) shows an example of a more general configuration. Recall that to minimize perspective aliasing, the shadow texel beams should be smaller than any pixel beams that they intersect. When the light is behind the camera, the narrowest pixel beams are encountered on the faces of the view frustum that face the light. To avoid aliasing, the shadow texels must become narrower in region A with increasing s. The shadow texel spacing is constant in region B because the minimum pixel beam width is the size of the pixels on the image plane, which is also constant. In region C the texel spacing begins to increase again. Previous shadow map warping algorithms use a single monotonic warping function for the entire frustum. In at least one of the sections, the texels spacing will grow in the opposite direction of the warping function, leading to high errors. Therefore, other warping algorithms reduce the amount of warping as the angle between the light and view directions becomes smaller. When they are parallel, a standard orthogonal projection is used. Frustum subdivision avoids the light direction problem by subdividing along the frustum faces (edges in 2D). Since a warp is applied to each subdivision independently the error is more tightly bounded.

We allocate shadow texels to each frustum subdivision according to the maximum error in each subdivision. If subdivision i has a maximum error of e_i and there are r_s texels in the shadow map, then the number of shadow texels allocated for each subdivision is:

$$r_{si} = r_s \frac{e_i}{\Sigma_j e_j}. \tag{6}$$

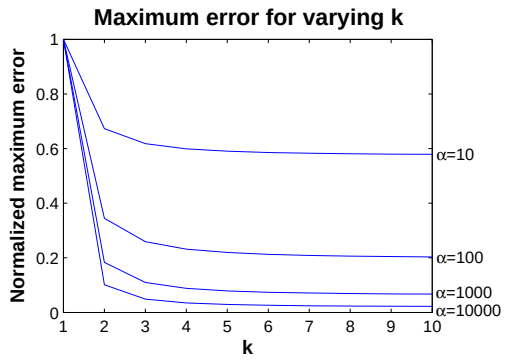

Figure 7: **Maximum error as a function of the number of subdivisions k for varying values of α.** *Each plot is normalized to show the overall shape. 2 or 3 subdivision bring the largest reduction in error.*

This allocation heuristic insures that the maximum error is spread evenly over all the subdivisions. If the same warping parameter ξ is used for all subdivisions, the heuristic also tends to allocate more texels to larger subdivisions. According to Equation 5, the maximum error with fixed ξ depends on α. For larger faces, α will be larger and the error greater.

Face subdivision. Once the frustum has been subdivided by its faces, further subdivisions along the faces create a better approximation of the optimal logarithmic transform, thus reducing error even more. If we subdivide along a face such that each subdivision is self-similar (as shown in Figure 6(b)), the maximum error in each subdivision will be the same. For each subdivision $i = 1, 2, ..., k$ we choose new near and far planes $[z_{i-1}, z_i]$:

$$z_i = \alpha^{i/k}.$$

With k subdivisions the ratio of the far to near plane distances, α_k, becomes $\alpha_k = \alpha_0^{1/k}$. According to texture allocation heuristic shown in Equation 6, each face subdivision i will be allotted $r_{si} = 1/k$ of the original shadow texels. Plugging in these values into Equation (5), the maximum error as a function of k becomes:

$$m_{\max} = k\frac{r_p}{r_s}\ \frac{(\alpha^{1/k}-1)}{2\tan\theta}\ \frac{(1+|\xi|(\sqrt{\alpha^{1/k}}-1))}{\sqrt{\alpha^{1/k}}}. \qquad (7)$$

Figure 7 shows how the maximum error changes with the number of face subdivisions k. Most of the error reductions come with only 2 or 3 subdivisions. This is important because we would like to keep the number of sections to minimum to avoid any extra rendering overhead.

4.1 Shadow warping and subdivision in 3D

So far we have only considered 2D scenes. Most of the machinery presented in the previous sections extends to 3D. As in 2D we first perform frustum subdivision. A frustum in general position, as shown in Figure 8(a), has up to 5 subdivisions defined by the faces, 4 sides and the near (or far) plane. If the edges shared by the sides and the near plane are shifted slightly to include the viewpoint, only 4 subdivision need to be used to cover the whole frustum. This introduces a small overlap between the subdivisions and only slightly more error. After frustum subdivision, we perform face subdivision. Next, each subdivision is transformed to a canonical space via a shear and a scale where the warping functions are computed (see Figure 8(b)). We use the same warping parameter, ξ, for each subdivision. Finally, the subdivisions are laid out in the shadow map (see Figure 8(c)).

Error in the x direction. There are two major differences with subdivided shadow maps in 2D and 3D. The first difference is that there is an added dimension in the shadow map, t, which corresponds to the x direction in Figure 3. The perspective projection applied to z also affects x. The error in x is optimal when $\xi = -1$, in which case the frustum of the perspective projection exactly matches that of the camera (as in PSMs). The error in z, however, is at its maximum with $\xi = -1$. Unfortunately, it is not possible to minimize the error in both the x and z directions at the same time.

We can follow a similar analysis for the x to obtain the maximum x error for a subdivision. We start with the width of a shadow texel beam in the t direction:

$$\begin{aligned} w_t &= \frac{2\alpha\tan\theta}{r_t}\ \frac{z'}{f} \\ &= \frac{2\alpha\tan\theta}{r_t}\ \frac{(z+n-1)}{(n+\alpha-1)}. \end{aligned}$$

The error function in x is then given by:

$$m_x = \frac{w_t}{w_p} = \frac{r_p}{r_t}\ \alpha\ \frac{(n+z-1)}{z(n+\alpha-1)}. \qquad (8)$$

The maximum x error always occurs on the near plane. Plugging in $z = 1$ and the expressions for n in Equation 4 we obtain for the maximum error in x:

$$m_{x\,\max} = \frac{r_p}{r_t}\ \alpha\ \begin{cases} \frac{1-\xi(\sqrt{\alpha}-1)}{\sqrt{\alpha}}, & \xi \le 0, \\ \frac{1}{\sqrt{\alpha}(1+\xi(\sqrt{\alpha}-1))}, & \xi > 0. \end{cases} \qquad (9)$$

Subdivision layout. The second major difference has to do with how the sections are laid out in the shadow map. We take advantage of the fact that the errors of each subdivision are correlated when the same warping parameter is used for all subdivisions. The maximum x errors is the same for subdivisions from opposite sides of the frustum (AC and BD) if the faces have been subdivided. For the maximum z error there is a trade off between opposite sides of the frustum. If the error is relatively high for one subdivision, it be low for its opposite subdivision. Based on these observations, we first split the shadow map in the t direction according to the maximum x error of the AC and BD pairs. Then we independently split in the s direction for subdivisions arising from each face pair according to maximum z error, e.g. between A and C.

5 Results

In this section we describe our implementation of subdivided shadow maps and highlight their performance on large models compared to other shadow map methods.

5.1 Implementation

We have implemented our algorithm on 2.4GHz Pentium-IV PC with 1GB RAM and a GeForce 7800 GTX with 256MB of video memory.

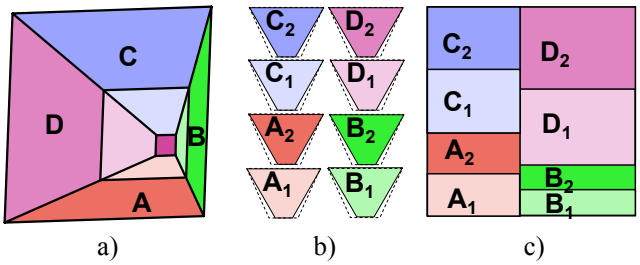

Figure 8: **Basic algorithm.** ***a)*** *Frustum and face subdivisions performed on faces of the eye view frustum as seen from the light.* ***b)*** *Each subdivision transformed to a canonical space and fitted with a warp (dashed frustum).* ***c)*** *Texture space layout according the maximum error in each subdivision.*

Model	Vert. (M)	Tri. (M)	Obj.
Power plant	11	12.2	1200
Double eagle tanker	77	82	3346
St. Matthew	186	372	1

Table 1: **Benchmark models:** *Model complexity of benchmark models is shown.*

Benchmark models. We tested out algorithm with three complex models, a coal-fired power plant composed of more than 12 million polygons and 1200 objects (Fig. 2), a double eagle tanker model consisting of 82 million polygon and 3346 objects (Fig. 5), the St. Matthew model consisting of a single 372 million polygon and single object (Fig. 11). The details of these models are shown in Table 1. We generated paths in each of our test models and used them to test the performance of our algorithm. These paths of the power plant and St. Matthew model are shown in the accompanying video.

Fragment program. We store the shadow maps for each subdivision in the same texture. In the fragment program we must be able to choose which shadow map to use at any pixel. We use the method described by Aldridge [2004]. We define planes parallel to the light through the edges of the visible faces. The normals are oriented facing left in the light's view. The dot product of a vertex with each of the edge planes is stored in separate channels of a texture coordinate and interpolated over the scene. For edges not visible to the light we set the dot product to 0. The following code will set to 1 only the channel corresponding to the face in the which a fragment lies, while setting all other channels to 0.

```
faceSelect = sign(dotProducts);
faceSelect = saturate(faceSelect * saturate(-faceSelect.yzwx));
```

Similarly we define split planes parallel to the light that pass through the edge induced by a face subdivision and track the dot product of a vertex with the split planes.

```
splitSelect = splitDotProducts >= 0;
```

The `faceSelect` and `splitSelect` are then combined to select the set of texture coordinate corresponding to the subdivision in which the fragment lies:

```
select0 = faceSelect * splitSelect;
select1 = faceSelect * (1.0 - splitSelect);
texCoord = select0.r * texCoord0 + select1.r * texCoord4 +
```

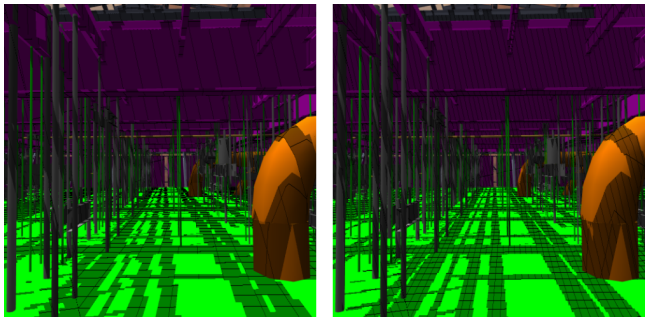

Figure 9: **Subdivided shadow maps.** *a) without and b) with face subdivision. For this scene with* $\alpha = 100$*, face subdivision leads to nearly* 5 *times improvement in maximum error.*

Model	St. Matthew	Power plant
SSM	1.16	1.67
SSM*	1.25	2.48

Table 2: **Overdraw Factors.** *Overdraw factors of the subdivided shadow maps with and without subdivisions (SSM and SSM*) are shown. Overdraw factors of the power plant model are much higher than those of St. Matthew since size and irregularity of objects of the power plant is much higher.*

```
select0.g * texCoord1 + select1.g * texCoord5 +
select0.b * texCoord2 + select1.b * texCoord6 +
select0.a * texCoord3 + select1.a * texCoord7;
```

Integration. Our method is simple to integrate with code that already uses shadow maps. Using functions we provide, the user first queries in which sections an object falls. In the subdivided shadow map render function we loop over each subdivision, setting up the view parameters and invoking a user supplied render callback. We supply the current subdivision as a parameter to the callback so that the user can render the objects which fall in this subdivisions. In this way our code does not need to know the details of the rest of the rendering system and the system does not need to know about how the shadow maps are handled.

Interactive shadow generation on complex models. To highlight the performance and high quality of our shadow maps on complex and massive models, we integrate our algorithm with out-of-core view-dependent rendering system [Yoon et al. 2004]. The rendering system uses a set of clusters decomposed from an input model as a scene representation of the model and computes progressive meshes to provide smooth level-of-detail transitions for each cluster at runtime. We also take advantage of visibility culling techniques [Govindaraju et al. 2003] to effectively cull away portion of the mesh that are not visible to the eye or the light.

5.2 Analysis and Limitation

For many of our comparisons we chose TSMs over PSMs and LSPSMs because TSMs gave slightly better quality.

The size of the objects in the scene has a great impact on the rendering time for the shadow map. Since each subdivision must be rendered into the shadow map separately,

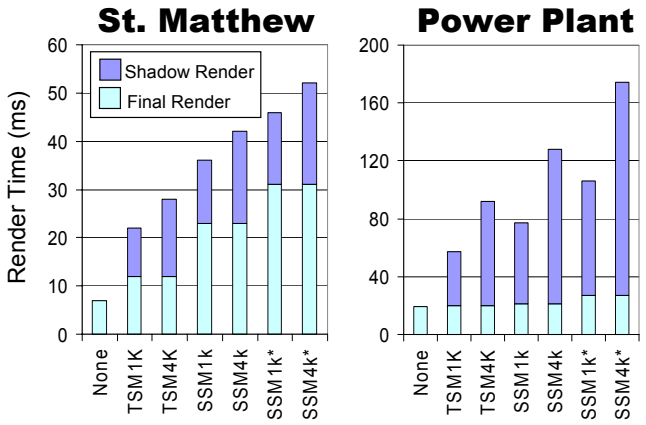

Figure 10: *Timing comparison between trapezoidal shadow maps (TSM) and subdivided shadow maps with and without face subdivisions (SSM* and SSM) at 1K×1K and 4K×4K resolution.*

objects that fall into more than one subdivision will be rendered multiple times. If a scene is decomposed into sub-objects that are sufficiently smaller than the subdivisions, most sub-objects will be rendered only once. Therefore, the rendering cost of transforming scene geometry will not increase much. The subdivided shadow map has the same resolution as a standard shadow map, so the fill-rate consumption will also about the same. But if the objects are large, they will span multiple subdivisions, hurting performance. We measure excessive rendering with an *overdraw factor*:

$$\text{overdraw} = \frac{\text{number of object renderings}}{\text{number of objects}}$$

The graph in Figure 10 shows that when more subdivisions are used, the shadow map rendering time for the power plant increases much more than for the statue. The reason for this is that the power plant has much more irregular geometry resulting in clusters that are larger that those of the statue. This leads to much higher overdraw factors (see Table 2).

The image rendering time for subdivided shadow maps is higher than for TSMs. This is due largely to the use of a more complicated fragment program for subdivided shadow maps. The fragment program that supports a face subdivision is even more complicated, leading to even longer render times. If the overdraw factor is low, the total frame time for subdivided shadow maps should be roughly twice that for a shadow warping algorithm. However, the quality may be much more than twice. Figure 1 shows a comparison of subdivided shadow maps with TSMs. For this view even the largest shadow map we could allocate, 4K×4K, was not sufficient to match the quality that we achieved with a 1K×1K subdivided shadow map.

The error reduction with subdivided shadow maps will be most dramatic when the angle between the light and view directions is small because other shadow warping algorithms do not handle this case well. When the light and view directions are perpendicular in the optimal configuration, frustum subdivision does not improve the quality much over the other shadow warping algorithms. A single set of face subdivisions, however, reduces the error quite a bit. Figure 9 SSMs with and without face subdivision.

Often the improvement in shadow quality from using face subdivision does not appear to be as good as predicted by

Figure 11: **St. Matthew model with shadows.** *This image shows St. Matthew model consisting of* 372 *million triangles. We are able to render the model with high quality shadows at* 15-25 *frames per second on commodity hardware with a GeForce 7800 GTX GPU.*

Equation 7. The equation predicts the maximum over an entire subdivision. If there are no surfaces where the error is at its maximum, the reduction of error will not be visible. The error is reduced in other parts of the scene as well, but not as much.

One problem with subdivided shadow maps is that the texels of the shadow map can be excessively sheared. The shear comes from the transformation of the subdivisions to the canonical space for warping. The texel shearing can be alleviated somewhat by first rotating the subdivision before shearing. This introduces more error but may yield more pleasing results.

Conclusion and Future Work

We have presented a shadow map algorithm for reducing perspective aliasing by applying separate warp functions to subdivisions of the scene individually. The algorithm can reduce the maximum aliasing by a factor of 10 or more while running only about twice as slow as other shadow map warping algorithms.

We would like to extend our technique to be used with point lights. The analysis is a bit more involved for point lights because now shadow texel beams change width with distance. Cube maps can be seen as a form of subdivision. We conjecture that warping the faces of the cube maps could lead to higher quality shadows.

References

Aila, T., and Laine, S. 2004. Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004*, Eurographics Association, 161–166.

Aldridge, G. 2004. Generalized trapezoidal shadow mapping for infinite directional lighting. *http://legion.gibbering.net/projectx/paper/shadow%20mapping/*.

Chan, E., and Durand, F. 2004. An efficient hybrid shadow rendering algorithm. In *Proceedings of the Eurographics Symposium on Rendering*, Eurographics Association, 185–195.

Chong, H., and Gortler, S. 2004. A lixel for every pixel. In *Proceedings of the Eurographics Symposium on Rendering*, Eurographics Association, 167–172.

Fernando, R., Fernandez, S., Bala, K., and Greenberg, D. 2001. Adaptive shadow maps. In *Proceedings of ACM SIGGRAPH 2001*, 387–390.

Forum. 2003. Shadow mapping in large environments. *http://www.gamedev.net/community/forums/topic.asp?topic_id=179941*.

Govindaraju, N., Lloyd, B., Yoon, S., Sud, A., and Manocha, D. 2003. Interactive shadow generation in complex environments. *Proc. of ACM SIGGRAPH/ACM Trans. on Graphics 22*, 3, 501–510.

Johnson, G., Mark, W., and Burns, C. 2004. The irregular z-buffer and its application to shadow mapping. In *The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-04-09*.

Kozlov, S. 2004. *Perspective Shadow Maps: Care and Feeding*. Addison-Wesley, 214–244.

Lefohn, A., Sengupta, S., Kniss, J. M., Strzodka, R., and Owens, J. D. 2005. Dynamic adaptive shadow maps on graphics hardware. In *ACM SIGGRAPH 2005 Conference Abstracts and Applications*.

Martin, T., and Tan, T.-S. 2004. Anti-aliasing and continuity with trapezoidal shadow maps. In *Proceedings of the Eurographics Symposium on Rendering*, Eurographics Association, 153–160.

McCool, M. 2000. Shadow volume reconstruction from depth maps. *ACM Trans. on Graphics 19*, 1, 1–26.

Segal, M., Korobkin, C., van Widenfelt, R., Foran, J., and Haeberli, P. 1992. Fast shadows and lighting effects using texture mapping. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, vol. 26, 249–252.

Sen, P., Cammarano, M., and Hanrahan, P. 2003. Shadow silhouette maps. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH) 22*.

Stamminger, M., and Drettakis, G. 2002. Perspective shadow maps. In *Proceedings of ACM SIGGRAPH 2002*, 557–562.

Tadamura, K., Qin, X., Jiao, G., and Nakamae, E. 1999. Rendering optimal solar shadows using plural sunlight depth buffers. In *Computer Graphics International 1999*, 166.

Williams, L. 1978. Casting curved shadows on curved surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, vol. 12, 270–274.

Wimmer, M., Scherzer, D., and Purgathofer, W. 2004. Light space perspective shadow maps. In *Proceedings of the Eurographics Symposium on Rendering*, Eurographics Association, 143–152.

Yoon, S.-E., Salomon, B., Gayle, R., and Manocha, D. 2004. Quick-VDR: Interactive View-dependent Rendering of Massive Models. *IEEE Visualization*, 131–138.

Appendix

A parameterization s using a general projective transform on z is given by:

$$s = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} z \\ 1 \end{bmatrix} = \begin{bmatrix} az+b \\ cz+d \end{bmatrix}$$

After the perspective divide we have a function:

$$\begin{aligned} s &= \frac{az+b}{cz+d} \\ \frac{ds}{dz} &= \frac{ad-bc}{(cz+d)^2} \end{aligned}$$

Thus the texel spacing in world space, dz/ds, generated by a perspective projection is proportional to z^2.