Logarithmic Perspective Shadow Maps

Technical Report TR07-005
University of North Carolina at Chapel Hill
June 2007

D. Brandon Lloyd[1] Naga K. Govindaraju[2] Cory Quammen[1] Steve Molnar[3] Dinesh Manocha[1]

[1] University of North Carolina at Chapel Hill [2] Microsoft Corporation [3] NVIDIA Corporation

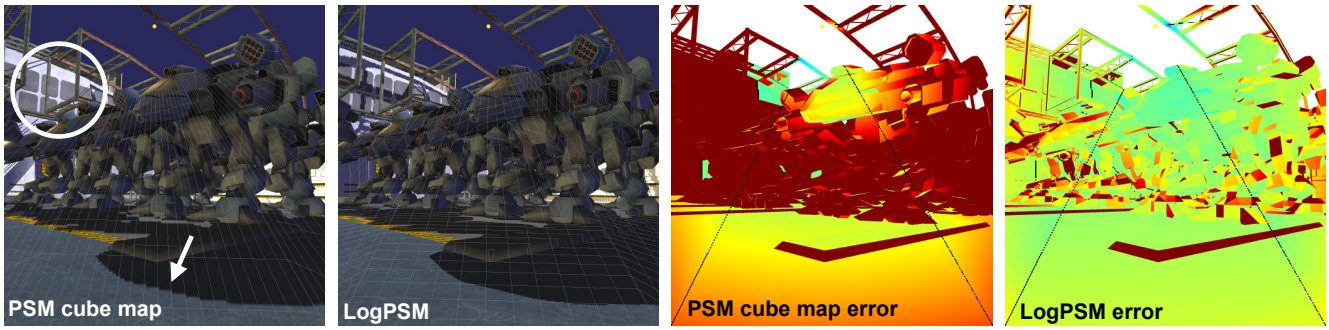

Figure 1: *Night-time scene of robots in a hangar with a point light. We compare our algorithm (LogPSM) to Kozlov's improved perspective shadow map (PSM) algorithm. Both algorithms use a cube map with a* total *resolution of* 1024×1024. *The images have a resolution of* 512×512. *(Left) Compared to a standard cube map, the PSM cube map greatly reduces aliasing artifacts near the viewer, but some aliasing is still visible. The shadows are severely stretched on the back wall. LogPSMs provide higher quality both near the viewer and in the distance. The shadow map grid has been superimposed to aid visualization (grid lines every 20 texels). (Right) An error visualization for both algorithms. We use an error metric m that is essentially the maximum extent of a shadow map texel projected into the image. Green represents no aliasing* ($m = 1$) *and dark red* ($m > 10$) *represents high aliasing. LogPSMs provide significantly lower maximum error and the error is more evenly distributed.*

Abstract

We present a novel shadow map parameterization to reduce perspective aliasing artifacts for both point and directional light sources. We derive the aliasing error equations for both types of light sources in general position. Using these equations we compute tight bounds on the aliasing error. From these bounds we derive our shadow map parameterization, which is a simple combination of a perspective projection with a logarithmic transformation. We then extend existing algorithms to formulate three types of logarithmic perspective shadow maps (LogPSMs) and analyze the error for each type. Compared with competing algorithms, LogPSMs can produce significantly less aliasing error. Equivalently, for the same error as competing algorithms, LogPSMs can produce significant savings in storage and bandwidth. We demonstrate the benefit of LogPSMs for several models of varying complexity.

1 Introduction

The shadow map algorithm [Williams 1978] is a popular approach for hard shadow computation in interactive applications. It uses a depth buffer rendered from the viewpoint of the light to compute shadows during image rendering. Shadow maps are relatively easy to implement, deliver good performance on commodity GPUs, and can handle complex, dynamic scenes. Other alternatives for hard shadows, like shadow volumes or ray tracing, can produce high-quality shadows, but may exhibit poor performance on complex models or dynamic scenes.

A major disadvantage of shadow maps is that they are prone to aliasing artifacts. Aliasing error can be classified as perspective or projection aliasing [Stamminger and Drettakis 2002]. Possible solutions to overcome both kinds of aliasing include using a high resolution shadow map, an adaptive shadow map that supports local increase in shadow map resolution, or an irregularly sampled shadow map. However, these techniques either require high memory bandwidth, are more complex to implement in the current pipeline, or require major changes to the current rasterization architectures.

Among the most practical shadow mapping solutions are those that reduce perspective aliasing by reparameterizing the shadow map to allocate more samples to the undersampled regions of a scene. Several warping algorithms, such as perspective shadow maps (PSMs) [Stamminger and Drettakis 2002] and their variants [Wimmer et al. 2004; Martin and Tan 2004], have been proposed to alter a shadow map's sample distribution. However, the parameterization used by these algorithms may produce a poor approximation to the optimal sample distribution. Therefore existing warping algorithms can still require high shadow map resolution to reduce aliasing. Recent work suggests that a logarithmic parameterization is closer to optimal [Wimmer et al.

2004; Lloyd et al. 2006; Zhang et al. 2006a]. Other algorithms produce a discrete approximation of a logarithmic parameterization by partitioning the view frustum along the view vector and using a separate shadow map for each subfrustum [Zhang et al. 2006a; Engel 2007]. These algorithms, however, generally require a large number of partitions to converge.

Main Results

In this paper, we present logarithmic perspective shadow maps (LogPSMs). LogPSMs are essentially extensions of an existing perspective warping algorithms. Our algorithms use a small number of partitions in combination with an improved warping function to achieve high performance with significantly less error than competing algorithms. Some of the novel aspects of our work include:

1. **Error analysis for general configurations.** We compute tight bounds on perspective aliasing error for point and directional lights in general configurations. The error analysis performed in previous work has typically been restricted to directional lights in a few special configurations [Stamminger and Drettakis 2002; Wimmer et al. 2004].

2. **LogPSM parameterization.** Based on this analysis we derive a new shadow map parameterization with tight bounds on the perspective aliasing error. We show that the error is $O(\log(f/n))$ where f/n is the ratio of the far to near plane distances of the view frustum. In contrast, the error of existing warping algorithms is $O(f/n)$. The parameterization is a simple combination of a logarithmic transformation with a perspective projection.

3. **LogPSM algorithms.** Our algorithms are the first to use a continuous logarithmic parameterization and can produce high quality shadows with less resolution than that required by other algorithms.

We perform a detailed comparison of LogPSMs with other algorithms using several different techniques. We demonstrate significant error reductions on different models of varying complexity.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents our derivation of the aliasing error equations. In Section 4 we use these equations to derive tight bounds on the error. In Section 5 we derive the LogPSM parameterization. Section 6 describes the various LogPSM algorithms. We present our results and comparisons with other algorithms in Section 7 and then conclude with some ideas for future work.

2 Previous Work

In this section we briefly review the approaches currently used to handle shadow map aliasing. The approaches can be classified as sample redistribution techniques, improved reconstruction techniques, and hybrid approaches. Besides shadow mapping, other algorithms for hard shadow generation include ray tracing and shadow volume algorithms. However, these approaches are unable to handle complex dynamic environments at high resolution and high frame rates.

2.1 Sample redistribution techniques

We categorize sample redistribution techniques according to the method that they use: warping, partitioning, combinations of warping and partitioning, and irregular sampling.

Warping. The use of warping to handle aliasing was introduced with perspective shadow maps (PSMs) [Stamminger and Drettakis 2002]. A PSM is created by rendering the scene in the post-perspective space of the camera. Light-space perspective shadow maps (LiSPSMs) [Wimmer et al. 2004] are a generalization of PSMs which avoid some of the problems of rendering in post-perspective space. Trapezoidal shadow maps (TSMs) [Martin and Tan 2004] are similar to LiSPSMs, except that the parameter for the perspective projection is computed differently. Chong and Gortler [2006] propose an optimization framework for computing a perspective projection that minimizes aliasing in the scene.

Partitioning. Plural sunlight buffers [Tadamura et al. 1999] and cascaded shadow maps [Engel 2007] use a simple partitioning approach that splits the frustum along the view vector at intervals that increase geometrically with distance from the eye. Adaptive shadow maps [Fernando et al. 2001] and resolution matched shadow maps [Lefohn 2006] represent the shadow map as a quadtree of fixed resolution tiles. While the quadtree-based approaches can deliver high quality, they may require dozens of render passes. Tiled shadow maps [Arvo 2004] partition a single, fixed-resolution shadow map into tiles of different sizes guided by an error measurement heuristic. Forsyth [2006] proposes a partitioning technique that uses a greedy clustering of objects into multiple shadow frusta.

Warping + partitioning. Chong [2003] presents an algorithm for 2D flatland which performs perspective warping combined with partitioning. Chong and Gortler [2004] use a general projective transform to establish a one-to-one correspondence between pixels in the image and the texels in the shadow map, but only for a single plane within the scene. They use a small number of shadow maps to cover a few prominent, planar surfaces. Kozlov [2004] uses a cube map in the post-perspective space of the camera. Parallel-split shadow maps (PSSM) [Zhang et al. 2006a] combine warping with a partitioning similar to that of cascaded shadow maps. Queried virtual shadow maps [Giegl and Wimmer 2007] combine LiSPSMs with an adaptive partitioning scheme. Lloyd et al. [2006] analyze various combinations of warping and partitioning.

Irregular sampling. Instead of inferring sample locations from a parameterization of a regular grid, irregular shadow maps [Johnson et al. 2004; Aila and Laine 2004] sample explicitly at the query locations generated during image rendering. The results are equivalent to ray tracing. Since graphics hardware is optimized for rasterizing and storing samples on a regular grid, irregular shadow maps can be difficult to implement efficiently on current graphics architectures. A hardware architecture to support irregular shadow maps has been proposed [Johnson et al. 2005], but requires significant modifications of current GPUs to support efficient data scattering.

2.2 Improved reconstruction techniques

Aliasing artifacts arise from the use of nearest-neighbor filtering when reconstructing the visibility "signal" from sampled information in the shadow map. Standard filtering techniques, such as interpolation, cannot be applied directly to a depth-based shadow map. Percentage closer filtering (PCF)

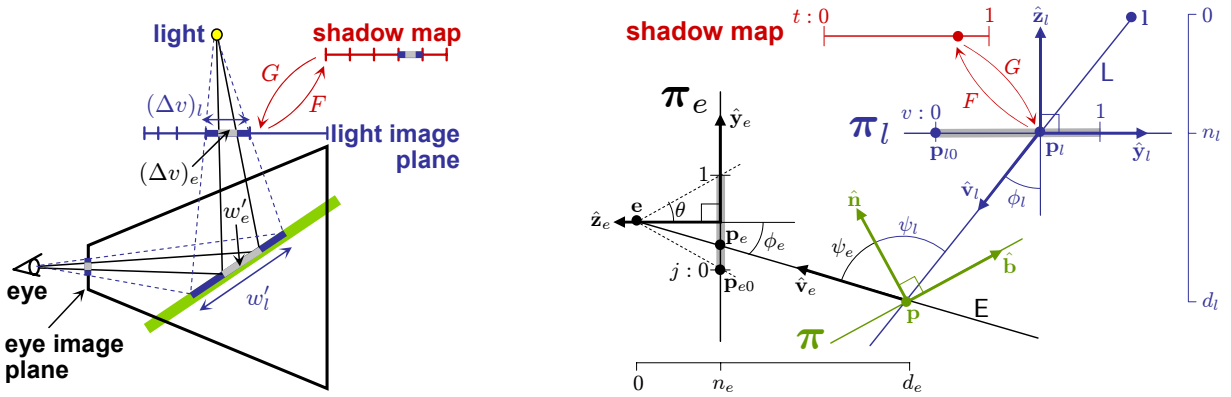

Figure 2: Computing aliasing error. *(Left) Sample locations corresponding to shadow texels and image pixels. Aliasing error can be quantified as the ratio of the spacing these shadow map and image sample locations. (Right) The sample spacing is related to the derivatives of the function that maps a point $t \in [0,1]$ in the shadow map to $\mathbf{p}_l$ on the light's image plane $\boldsymbol{\pi}_l$ and projects it first through the light onto a planar surface $\boldsymbol{\pi}$ and then through the eye to a point $\mathbf{p}_e$ on the eye's image plane $\boldsymbol{\pi}_e$.*

[Reeves et al. 1987] first computes visibility at sample locations and then applies a filter to the results. Variance shadow maps [Donnelly and Lauritzen 2006] store a statistical representation of depth to which standard texture filtering techniques can be applied. Both of these improved filtering techniques replace jagged reconstruction errors with a blurred shadow edge. Silhouette shadow maps [Sen et al. 2003] reconstruct a hard edge by augmenting the shadow map with extra information about the location of the silhouette edges of shadow casters.

2.3 Hybrid approaches

Hybrid techniques combine shadow maps with object-space methods. The shadow volume algorithm [Crow 1977] is a popular object-space method that does not suffer from aliasing artifacts but does require high fill rate and memory bandwidth. Shadow maps have been combined with shadow volumes to address the fill problems. McCool [2001] constructs a shadow map from edges in a shadow map, yielding a simplified shadow volume. Chan and Durand [2004] use a shadow map to mask off shadow volume rendering in regions that are not near shadow boundaries.

3 Shadow Map Aliasing Error

In this section we describe how to quantify aliasing error and derive the equations for it for a 2D scene. To our knowledge, ours is the first analysis that quantifies perspective aliasing error for point lights in general position. The analysis also extends to directional lights. The analysis in previous work is typically performed for a directional light for a few specific configurations [Stamminger and Drettakis 2002; Wimmer et al. 2004; Lloyd et al. 2006]. A notable exception [Zhang et al. 2006b] computes perspective aliasing error for directional lights over a range of angles, but only along a single line through the view frustum. We then extend our 2D analysis to 3D.

3.1 Quantifying aliasing error

Aliasing error is caused by a mismatch between the sample locations used to render the shadow map from the viewpoint of the light and the image from the viewpoint of the eye. Ideally, the eye samples would correspond exactly with the light samples, as with the raytracing or the irregular z-buffer. Shadow map warping methods instead seek to match sampling rates. For our derivations we choose to work with the spacing between samples since it is geometrically more intuitive. Figure 2 shows the sample spacing corresponding to eye and light samples at various locations in a simple 2D scene. Aliasing error occurs when the light sample spacing is greater than the eye sample spacing. If $j \in [0,1]$ and $t \in [0,1]$ are normalized image and shadow map coordinates, then aliasing error can be quantified as:

$$m = \frac{r_j}{r_t}\frac{\mathrm{d}j}{\mathrm{d}t}, \tag{1}$$

where r_j and r_t are the image and shadow map resolutions. (The choice of j and t as coordinates is due to the fact that we are essentially looking at the side view of a 3D frustum. We will also use the coordinates i and s later when we look at the equations for 3D). Aliasing occurs when $m > 1$.

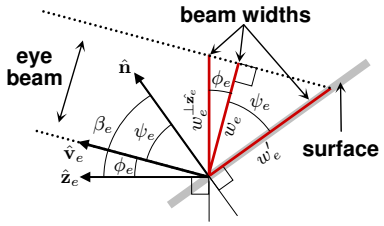

Figure 3: Computing the projected eye beam width on a surface. *w_e is the actual width of the beam, w'_e is the width of the projection onto the surface, and $w^{\perp \hat{\mathbf{z}}_e}$ is the width of beam measured perpendicular to $\hat{\mathbf{z}}_e$.*

3.2 Deriving aliasing error

To derive $\mathrm{d}j/\mathrm{d}t$ in Equation 1 we compute the function $j(t)$ from Figure 2 by tracing a sample from the shadow map to its corresponding location in the image. We begin by introducing our notation. A point $\mathbf{p}$ and a vector $\vec{\mathbf{v}}$ are expressed as a column vectors in affine coordinates with the last entry equal to 1 for a point and 0 for a vector. Simpler formulas might be obtained by using full homogeneous coordinates, but we are interested in an intuitive definition of aliasing in terms of distances and angles, which are easier to compute

with affine coordinates. $\hat{\mathbf{v}}$ is a normalized vector. A plane (or line in 2D) $\boldsymbol{\pi}$ is a row vector $(\hat{\mathbf{n}}^\top, -D)$, where D is the distance to the plane (line) from the origin along the normal $\hat{\mathbf{n}}$. $\boldsymbol{\pi}\mathbf{p}$ gives the signed distance of $\mathbf{p}$ from the plane and $\boldsymbol{\pi}\hat{\mathbf{v}}$ gives $\hat{\mathbf{n}} \cdot \hat{\mathbf{v}} = \cos\theta$, where θ is the angle between $\hat{\mathbf{n}}$ and $\hat{\mathbf{v}}$.

A function G, the inverse of the shadow map parameterization F, maps a point t in the shadow map to the normalized light plane coordinate $v \in [0, 1]$. The point $\mathbf{p}_l$ on the light image plane $\boldsymbol{\pi}_l$ corresponding to v is given by:

$$\mathbf{p}_l = \mathbf{p}_{l0} + vW_l\hat{\mathbf{y}}_l, \tag{2}$$

where W_l is the width of the portion of $\boldsymbol{\pi}_l$ covered by the shadow map. Projecting $\mathbf{p}_l$ onto a planar surface $\boldsymbol{\pi}$ along the line L through the light position $\mathbf{l}$ yields:

$$\mathbf{p} = \mathbf{p}_l - \frac{\boldsymbol{\pi}\mathbf{p}_l}{\boldsymbol{\pi}(\mathbf{p}_l - \mathbf{l})}(\mathbf{p}_l - \mathbf{l}). \tag{3}$$

This point is then projected onto the eye image plane $\boldsymbol{\pi}_e$ along the line E through the eye position $\mathbf{e}$:

$$\mathbf{p}_e = \mathbf{p} - \frac{\boldsymbol{\pi}_e\mathbf{p}}{\boldsymbol{\pi}_e(\mathbf{e} - \mathbf{p})}(\mathbf{e} - \mathbf{p}). \tag{4}$$

The eye image plane is parameterized similarly to Equation 2 using j instead of v. The j coordinate can be computed from $\mathbf{p}_e$ as:

$$j = \frac{\hat{\mathbf{y}}_e \cdot (\mathbf{p}_e - \mathbf{p}_{e0})}{W_e} = \frac{\hat{\mathbf{y}}_e^\top(\mathbf{p}_e - \mathbf{p}_{e0})}{W_e}. \tag{5}$$

To compute $\mathrm{d}j/\mathrm{d}t$ we use the chain rule:

$$\frac{\mathrm{d}j}{\mathrm{d}t} = \frac{\partial j}{\partial \mathbf{p}_e}\frac{\partial \mathbf{p}_e}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \mathbf{p}_l}\frac{\partial \mathbf{p}_l}{\partial v}\frac{\mathrm{d}v}{\mathrm{d}t} \tag{6}$$

$$\frac{\partial j}{\partial \mathbf{p}_e} = \frac{\hat{\mathbf{y}}_e^\top}{W_e} \tag{7}$$

$$\frac{\partial \mathbf{p}_e}{\partial \mathbf{p}} = \mathbf{I} + \frac{\boldsymbol{\pi}_e\mathbf{p}}{\boldsymbol{\pi}_e(\mathbf{e} - \mathbf{p})}\mathbf{I} - \frac{\boldsymbol{\pi}_e\mathbf{e}}{(\boldsymbol{\pi}_e(\mathbf{c_e} - \mathbf{p}))^2}\left((\mathbf{e} - \mathbf{p})\boldsymbol{\pi}_e\right) \tag{8}$$

$$\frac{\partial \mathbf{p}}{\partial \mathbf{p}_l} = \mathbf{I} - \frac{\boldsymbol{\pi}\mathbf{p}_l}{\boldsymbol{\pi}(\mathbf{p}_l - \mathbf{l})}\mathbf{I} + \frac{\boldsymbol{\pi}\mathbf{l}}{(\boldsymbol{\pi}(\mathbf{p}_l - \mathbf{l}))^2}\left((\mathbf{p}_l - \mathbf{l})\boldsymbol{\pi}\right) \tag{9}$$

$$\frac{\partial \mathbf{p}_l}{\partial v} = W_l\hat{\mathbf{y}}_l \tag{10}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{\mathrm{d}G}{\mathrm{d}t}. \tag{11}$$

Equations 7 – 10 are expressions for Jacobian matrices. The derivative $\mathrm{d}j/\mathrm{d}t$ can be reduced to a simpler form. We first multiply together Equations 9–11:

$$\begin{aligned}\frac{\partial \mathbf{p}}{\partial t} &= \frac{\partial \mathbf{p}}{\partial \mathbf{p}_l}\frac{\partial \mathbf{p}_l}{\partial v}\frac{\partial v}{\partial t} \\ &= W_l\frac{\mathrm{d}G}{\mathrm{d}t}\frac{\boldsymbol{\pi}\mathbf{l}}{\boldsymbol{\pi}(\mathbf{p}_l - \mathbf{l})}\left(\hat{\mathbf{y}}_l - \frac{\boldsymbol{\pi}\hat{\mathbf{y}}_l}{\boldsymbol{\pi}(\mathbf{p}_l - \mathbf{l})}(\mathbf{p}_l - \mathbf{l})\right).\end{aligned} \tag{12}$$

The $\boldsymbol{\pi}\mathbf{l}$ and $\boldsymbol{\pi}(\mathbf{p}_l - \mathbf{l})$ terms are proportional to d_l and n_l, respectively, so their ratio can be replaced with d_l/n_l. The angle between $\hat{\mathbf{z}}_l$ and $\hat{\mathbf{n}}$ is β_l. Therefore $\boldsymbol{\pi}\hat{\mathbf{y}}_l = -\sin\beta_l$. We substitute $(\mathbf{p}_l - \mathbf{l}) = ||\mathbf{p}_l - \mathbf{l}||\hat{\mathbf{v}}_l$, where $-\hat{\mathbf{v}}$ is the light direction vector. The $||\mathbf{p}_l - \mathbf{l}||$ terms cancel leaving only $\hat{\mathbf{v}}$. We then replace $\boldsymbol{\pi}\hat{\mathbf{v}}$ with $-\cos\psi_l$:

$$\frac{\partial \mathbf{p}}{\partial t} = W_l\frac{\mathrm{d}G}{\mathrm{d}t}\frac{d_l}{n_l}\left(\hat{\mathbf{y}}_l - \frac{\sin\beta_l}{\cos\psi_l}\hat{\mathbf{v}}_l)\right). \tag{13}$$

m	aliasing error
$\tilde{m}$	perspective aliasing error
$\widetilde{M}$	maximum $\tilde{m}$ along a line L through the light
F, G	shadow map parameterization and its inverse
(i, j)	eye image plane coordinates
(s, t)	shadow map coordinates
(u, v)	light image plane coordinates
θ	frustum field of view
ϕ	angle between beam and image plane normal
ψ	angle between beam and surface normal
β	angle between image plane and surface normals
δ	spacing distribution function
$\tilde{\delta}_e$	perspective factor
δ_b	a lower bound on $\tilde{\delta}_e$
$r_i \times r_j$	image resolution
$r_s \times r_t$	shadow map resolution
ρ	parameterization normalizing constant
R	critical resolution factor
S	storage factor
R	maximum R over all light positions
S	maximum S over all light positions
γ	angle between light and view directions

Table 1: Important symbols used in this paper

We then expand $\hat{\mathbf{y}}_l$ an $\hat{\mathbf{v}}_l$ in terms of $\hat{\mathbf{n}}$ and $\hat{\mathbf{b}}$:

$$\hat{\mathbf{y}}_l = \cos\beta_l\hat{\mathbf{b}} - \sin\beta_l\hat{\mathbf{n}} \tag{14}$$

$$\hat{\mathbf{v}}_l = -\sin\psi_l\hat{\mathbf{b}} - \cos\psi_l\hat{\mathbf{n}}. \tag{15}$$

Substituting these equations into Equation 13 and utilizing the fact that $\psi_l - \beta_l = \phi_l$ yields:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} &= W_l\frac{\mathrm{d}G}{\mathrm{d}t}\frac{d_l}{n_l}\frac{(\cos\psi_l\cos\beta_l + \sin\psi_l\sin\beta_l)}{\cos\psi_l}\hat{\mathbf{b}} \\ &= W_l\frac{\mathrm{d}G}{\mathrm{d}t}\frac{d_l}{n_l}\frac{\cos(\psi_l - \beta_l)}{\cos\psi_l}\hat{\mathbf{b}} \\ &= W_l\frac{\mathrm{d}G}{\mathrm{d}t}\frac{d_l}{n_l}\frac{\cos\phi_l}{\cos\psi_l}\hat{\mathbf{b}}.\end{aligned} \tag{16}$$

Now we multiply together Equations 7, 8, and 16 and substitute $\boldsymbol{\pi}_e\mathbf{e} = n_e$, $\boldsymbol{\pi}_e(\mathbf{e} - \mathbf{p}) = d_e$, and $(\mathbf{e} - \mathbf{p}) = ||\mathbf{e} - \mathbf{p}||\hat{\mathbf{v}}_e$:

$$\frac{\mathrm{d}j}{\mathrm{d}t} = \frac{W_l}{W_e}\frac{n_e}{n_l}\frac{d_l}{d_e}\left(\hat{\mathbf{y}}_e^\top\hat{\mathbf{b}} - \frac{(\hat{\mathbf{y}}_e^\top\hat{\mathbf{v}}_e)(\boldsymbol{\pi}_e\hat{\mathbf{b}})}{\boldsymbol{\pi}_e\hat{\mathbf{v}}_e}\right). \tag{17}$$

Substituting $\hat{\mathbf{y}}_e^\top\hat{\mathbf{b}} = \cos\beta_e$, $\hat{\mathbf{y}}_e^\top\hat{\mathbf{v}}_e = \sin\phi_e$, $\boldsymbol{\pi}_e\hat{\mathbf{b}} = -\sin\beta_e$, and $\boldsymbol{\pi}_e\hat{\mathbf{v}}_e = \cos\phi_e$ and utilizing the fact that $\beta_e - \phi_e = \psi_e$ yields the simplified version of $\mathrm{d}j/\mathrm{d}t$:

$$\begin{aligned}\frac{\mathrm{d}j}{\mathrm{d}t} &= \frac{W_l}{W_e}\frac{n_e}{n_l}\frac{d_l}{d_e}\frac{(\cos\phi_e\cos\beta_e + \sin\phi_e\sin\beta_e)}{\cos\phi_e} \\ &= \frac{W_l}{W_e}\frac{n_e}{n_l}\frac{d_l}{d_e}\frac{\cos(\beta_e - \phi_e)}{\cos\phi_e} \\ &= \frac{W_l}{W_e}\frac{n_e}{n_l}\frac{d_l}{d_e}\frac{\cos\psi_e}{\cos\phi_e}.\end{aligned} \tag{18}$$

Plugging $\mathrm{d}j/\mathrm{d}t$ into Equation 1 yields the final expression for aliasing error:

$$m = \frac{r_j}{r_t}\frac{\mathrm{d}G}{\mathrm{d}t}\frac{W_l}{W_e}\frac{n_e}{n_l}\frac{d_l}{d_e}\frac{\cos\phi_l}{\cos\phi_e}\frac{\cos\psi_e}{\cos\psi_l}. \tag{19}$$

Intuitive derivation. Some basic intuition for the terms in Equation 19 can be obtained by considering an equivalent but less rigorous derivation of m from the relative sample spacing on a surface in the scene (see Figure 2). A beam from the light through a region on the light image plane corresponding to a single shadow map texel projects onto the surface with width w'_l. A beam from the eye through a pixel also projects onto the surface with width w'_e. The aliasing error on the surface is given by the ratio of the projected beam widths:

$$m = \frac{w'_l}{w'_e}. \tag{20}$$

By the properties of similar triangles, the width of a pixel on $\boldsymbol{\pi}_e$ at a distance of n_e from the eye becomes $w_e^{\perp \hat{\mathbf{z}}_e}$ at a distance of d_e where the beam intersects the surface:

$$w_e^{\perp \hat{\mathbf{z}}_e} = \frac{W_e}{r_j}\frac{d_e}{n_e} \tag{21}$$

For a narrow beam, we can assume that the sides of the beam are essentially parallel. From Figure 3 we see that multiplying $w_e^{\perp \hat{\mathbf{z}}_e}$ by $\cos\phi_e$ gives the actual width of the beam w_e and dividing by $\cos\psi_e$ gives the width of its projection w'_e:

$$w_e = w_e^{\perp \hat{\mathbf{z}}_e}\cos\phi_e \tag{22}$$

$$w'_e = \frac{w_e}{\cos\psi_e} = \frac{W_e}{r_j}\frac{d_e}{n_e}\frac{\cos\phi_e}{\cos\psi_e}. \tag{23}$$

Similarly, a shadow map texel maps to a segment of width $(W_l/r_t)(\mathrm{d}G/\mathrm{d}t)$ on the light image plane producing a projected light beam width on the surface of:

$$w'_l = \frac{W_l}{r_t}\frac{\mathrm{d}G}{\mathrm{d}t}\frac{d_l}{n_l}\frac{\cos\phi_l}{\cos\psi_l}. \tag{24}$$

Plugging Equations 23 and 24 into Equation 20 yields Equation 19.

Directional lights. As a point light at $\mathbf{l}$ moves away towards infinity along a direction $\hat{\mathbf{l}}$ it becomes a directional light. Equation 3 converges to:

$$\mathbf{p} = \mathbf{p}_l - \frac{\boldsymbol{\pi}\mathbf{p}_l}{\boldsymbol{\pi}\hat{\mathbf{l}}}\hat{\mathbf{l}}. \tag{25}$$

Equation 9 becomes:

$$\frac{\partial \mathbf{p}}{\partial \mathbf{p}_l} = \mathbf{I} - \frac{1}{\boldsymbol{\pi}\hat{\mathbf{l}}}\,\hat{\mathbf{l}}\boldsymbol{\pi}. \tag{26}$$

In Equation 19, the n_l/d_l term converges to 1 and the $\cos\phi_l$ term becomes constant.

The formulation for m in Equation 19 is similar to those used for aliasing error in previous work [Stamminger and Drettakis 2002; Wimmer et al. 2004; Zhang et al. 2006b]. However, our formulation is more general because it is valid for both point and directional lights and it takes into account the variation of eye and light beam widths as a function of ϕ_e and ϕ_l, respectively.

3.3 Factoring aliasing error

Our goal is to compute tight bounds on the aliasing error m which we can then use to formulate a low-error shadow map parameterization F. For convenience we split m into two main parts:

$$m = \frac{(\Delta v)_l}{(\Delta v)_e} \tag{27}$$

$$(\Delta v)_l = \frac{1}{r_t}\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{1}{r_t}\delta_l$$

$$(\Delta v)_e = \frac{1}{r_j}\frac{\mathrm{d}v}{\mathrm{d}j} = \frac{1}{r_j}\delta_e$$

$$\delta_l = \frac{\mathrm{d}G}{\mathrm{d}t} = \left(\frac{\mathrm{d}F}{\mathrm{d}v}\right)^{-1} \tag{28}$$

$$\delta_e = \tilde{\delta}_e\frac{\cos\psi_l}{\cos\psi_e} \tag{29}$$

$$\tilde{\delta}_e = \frac{W_e}{W_l}\frac{n_l}{n_e}\frac{d_e}{d_l}\frac{\cos\phi_e}{\cos\phi_l}. \tag{30}$$

Intuitively, $(\Delta v)_l$ and $(\Delta v)_e$ are the spacing in v between the corresponding light and eye samples (see Figure 2). This factorization is convenient because $(\Delta v)_e$ encapsulates all of the geometric terms while $(\Delta v)_l$ encapsulates the two factors that can be manipulated to control aliasing – the parameterization which determines the aliasing distribution and the shadow map resolution that controls the overall scale. (For a point light the orientation of the light image plane also affects the distribution of light samples and thus the aliasing error. We make a distinction between the light image plane and the near plane of the light frustum used to render the shadow map. The near plane is typically chosen based on other considerations, such as properly enclosing the scene geometry in the light frustum. Because one light image plane is related to another by a projective transformation that can be absorbed into the parameterization, we can use a "standard" light image plane that is convenient for calculation.) The spacing functions depend on a resolution factor and the *spacing distribution functions*, δ_l and δ_e, which are simply the derivatives $\mathrm{d}v/\mathrm{d}t$ and $\mathrm{d}v/\mathrm{d}j$, respectively. Because we wish to derive the parameterization F we have expressed δ_l in terms of F instead of its inverse.

Following Stamminger and Drettakis [2002], δ_e can be factored into two components – a *perspective factor*, $\tilde{\delta}_e$, and a *projection factor*, $\cos\psi_e/\cos\psi_l$. The perspective factor depends only on the position of the light relative to the view frustum and is bounded over all points inside the view frustum. The projection factor, on the other hand, depends on the orientation of the surfaces in the scene, and is potentially unbounded. In order to obtain a simple parameterization amenable to real-time rendering without incurring the cost of a complex, scene-dependent analysis, many algorithms ignore the projective factor and address only perspective aliasing error $\tilde{m}$:

$$\tilde{m} = \frac{r_j}{r_t}\frac{\delta_l}{\tilde{\delta}_e} = \frac{w_e}{w_l}. \tag{31}$$

$\tilde{m}$ can be thought of as measuring the ratio of the widths of the light and eye beam at a point. Alternatively, $\tilde{m}$ can be thought of as the aliasing error on a surface with a normal that is located half-way between the eye and light directions $\hat{\mathbf{v}}_e$ and $-\hat{\mathbf{v}}_l$. For such a surface the projection factor is 1.

3.4 Aliasing error in 3D

So far we have only analyzed aliasing error in 2D. In 3D we parameterize the eye's image plane, the light's image plane,

and the shadow map by the 2D coordinates $\mathbf{i} = (i, j)^\top$, $\mathbf{u} = (u, v)^\top$, and $\mathbf{s} = (s, t)^\top$, respectively. Each coordinate is in the range $[0, 1]$. Equation 2 becomes:

$$\mathbf{p}_l = \mathbf{p}_{l0} + uW_{lx}\hat{\mathbf{x}}_l + vW_{ly}\hat{\mathbf{y}}_l. \tag{32}$$

Equations 3 and 4 remain the same. We replace W_e in Equation 5 with the direction-specific W_{ey} and add the equation to compute i:

$$i = \frac{\hat{\mathbf{x}}_e^\top(\mathbf{p}_e - \mathbf{p}_{e0})}{W_{ex}}. \tag{33}$$

The light image plane parameterization is now a 2D function $\mathbf{u} = \mathbf{G}(\mathbf{s})$. With these changes the projection of a shadow map texel in the image is now described by a 2×2 aliasing matrix $\mathbf{M}_a$:

$$\mathbf{M}_a = \begin{bmatrix} r_i & 0 \\ 0 & r_j \end{bmatrix} \frac{\partial \mathbf{i}}{\partial \mathbf{s}} \begin{bmatrix} \frac{1}{r_s} & 0 \\ 0 & \frac{1}{r_t} \end{bmatrix} \tag{34}$$

$$\frac{\partial \mathbf{i}}{\partial \mathbf{s}} = \begin{bmatrix} \frac{\partial i}{\partial s} & \frac{\partial i}{\partial t} \\ \frac{\partial j}{\partial s} & \frac{\partial j}{\partial t} \end{bmatrix} = \frac{\partial \mathbf{i}}{\partial \mathbf{p}_e} \frac{\partial \mathbf{p}_e}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \mathbf{p}_l} \frac{\partial \mathbf{p}_l}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{s}} \tag{35}$$

$$\frac{\partial \mathbf{i}}{\partial \mathbf{p}_e} = \begin{bmatrix} \frac{\hat{\mathbf{x}}_e^\top}{W_{ex}} \\ \frac{\hat{\mathbf{y}}_e^\top}{W_{ey}} \end{bmatrix} \tag{36}$$

$$\frac{\partial \mathbf{p}_l}{\partial \mathbf{u}} = \begin{bmatrix} W_{lx}\hat{\mathbf{x}}_l & W_{ly}\hat{\mathbf{y}}_l \end{bmatrix} \tag{37}$$

$$\frac{\partial \mathbf{u}}{\partial \mathbf{s}} = \frac{\partial \mathbf{G}}{\partial \mathbf{s}}. \tag{38}$$

To obtain a scalar measure of aliasing error it is necessary to define a metric h that is a function of the elements of $\mathbf{M}_a$. Some possibilities for h include a matrix norm, such as a p-norm or the Frobenius norm, or the determinant, which approximates the area of the projected shadow map texel in the image.

In 3D the spacing distribution functions $\boldsymbol{\delta}_l$ and $\boldsymbol{\delta}_e$ are 2×2 Jacobian matrices:

$$\boldsymbol{\delta}_l = \frac{\partial \mathbf{G}}{\partial \mathbf{s}} = \left(\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\right)^{-1} \tag{39}$$

$$\boldsymbol{\delta}_e = \left(\frac{\partial \mathbf{i}}{\partial \mathbf{u}}\right)^{-1}. \tag{40}$$

The factorization of $\boldsymbol{\delta}_e$ into perspective and projective factors is not as straightforward as in 2D. Several possibilities exist. Based on the intuition of the 2D perspective error, one approach might be to compute the differential cross sections $\mathbf{X}_e$ and $\mathbf{X}_l$ of the light and eye beam at a point and define perspective aliasing error as the ratio $\tilde{m} = h(\mathbf{X}_l)/h(\mathbf{X}_e)$. Another possibility would be to take $\tilde{m} = h(\mathbf{M}_a)$ where the normal of the planar surface used to compute $\mathbf{M}_a$ is oriented half-way between $\hat{\mathbf{v}}_e$ and $-\hat{\mathbf{v}}_l$. But unlike the 2D case, these two approaches are not guaranteed to be equivalent.

4 Computing a parameterization with tight bounds on perspective aliasing error

We can control perspective aliasing error by modifying the sample spacing on the light image plane. The sample spacing is controlled by the resolution of the shadow map and the parameterization F. In this section we discuss how to compute F and in 2D from a given sample distribution function δ. We then compute a spacing distribution δ_b that produces tight bounds on the perspective aliasing error. This naturally leads to a partitioning of the shadow map into regions corresponding to the view frustum faces. Finally, we extend the analysis to 3D. Unfortunately, the parameterizations based on δ_b are too complicated for practical use, but they provide a good baseline for evaluating simpler parameterizations in the next section.

4.1 Computing a parameterization in 2D

Given a spacing distribution function $\delta(v)$, the parameterization F that produces $\delta_l \sim \delta(v)$ can be computed from Equation 28:

$$\left(\frac{\mathrm{d}F}{\mathrm{d}v}\right)^{-1} \sim \delta(v)$$

$$\frac{\mathrm{d}F}{\mathrm{d}v} = \frac{1}{\rho\,\delta(v)} \tag{41}$$

$$F(v) = \frac{1}{\rho}\int_0^v \frac{1}{\delta(\nu)}\mathrm{d}\nu. \tag{42}$$

ρ is the constant of proportionality. Normalizing F to the range $[0, 1]$ gives:

$$\rho = \int_0^1 \frac{1}{\delta(v)}\mathrm{d}v. \tag{43}$$

This process will work with any $\delta(v)$ so long as it is positive over the domain of integration.

4.2 Tight bounds on perspective aliasing error

Ideally we would like to ensure that $m = 1$ everywhere in the scene, thus eliminating aliasing while using the least amount of shadow map resolution. From Equation 27 we can see that $m = 1$ implies that $\delta_l(v) \sim \delta_e(v)$ and $r_t = \rho_e r_j$. In a scene with multiple surfaces, δ_e might not be expressible as a function of v. Along any line $\mathsf{L}(v)$ there may be a set of multiple points $\mathcal{P}$ that are visible to the eye, each with a different value of δ_e. One way to handle this is compute a lower bound on δ_e:

$$\delta_{e,\min}(v) = \min_{\mathbf{p}\in\mathcal{P}(v)}(\delta_e(\mathbf{p})). \tag{44}$$

Because $\delta_{e,\min}$ depends on scene geometry it can be arbitrarily complex. A lower bound on the perspective factor $\tilde{\delta}_e$ yields a simpler, scene-independent function:

$$\delta_b = \min_{\mathbf{p}\in\{\mathsf{L}(v)\cap\mathsf{V}\}}(\tilde{\delta}_e(\mathbf{p})). \tag{45}$$

V is the set of points inside the view frustum. Using δ_b we can define a tight upper bound on the perspective aliasing error $\widetilde{M}$ expressed as a function of v:

$$\widetilde{M}(v) = \max_{\mathbf{p}\in\{\mathsf{L}(v)\cap\mathsf{V}\}}(\tilde{m}(\mathbf{p})) = \frac{r_j}{r_t}\frac{\delta_l(v)}{\delta_b(v)}. \tag{46}$$

For a given $\delta_l(v)$, the resolution required to guarantee that there is no perspective aliasing in the view frustum, i.e. $\max_v(\widetilde{M}) = 1$, is $r_t = R_t r_j$, where R_t is given by:

$$R_t = \max_v\left(\frac{\delta_l(v)}{\delta_b(v)}\right) \tag{47}$$

We call R_t the *critical resolution factor*. R_t is the smallest when $\delta_l \sim \delta_b$, in which case $R_t = \rho_b$.

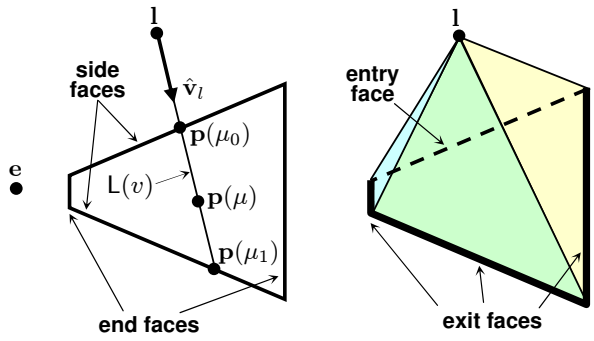

Figure 4: Face partitioning. *(a) The minimum perspective factor $\tilde{\delta}_e$ along a line segment L through the view frustum can be tightly bounded using a function whose minimum value occurs either at the faces where L enters or exits the frustum. (b) Based on this observation we can bound the perspective aliasing error by using a separate the shadow map for the region corresponding to the appropriate view frustum faces. For this light position, we use the exit faces.*

4.3 Computing δ_b

The points in the view frustum along a line $\mathsf{L}(v)$ can be parameterized as shown in Figure 4a:

$$\mathbf{p}(\mu) = \mathbf{l} + \mu\hat{\mathbf{v}}_l(v). \tag{48}$$

From Equation 30 we compute $\tilde{\delta}_e(\mu)$:

$$\tilde{\delta}_e(\mu) = c\frac{d_e(\mu)}{d_l(\mu)}\cos\phi_e(\mu) \tag{49}$$

$$d_e(\mu) = -\hat{\mathbf{z}}_e \cdot (\mathbf{p}(\mu) - \mathbf{e}) = -\mu(\hat{\mathbf{z}}_e \cdot \hat{\mathbf{v}}_l) - \hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e}) \tag{50}$$

$$d_l(\mu) = -\hat{\mathbf{z}}_l \cdot (\mathbf{p}(\mu) - \mathbf{l}) = -\mu(\hat{\mathbf{z}}_l \cdot \hat{\mathbf{v}}_l) \tag{51}$$

$$\cos\phi_e(\mu) = \frac{d_e(\mu)}{||\mathbf{p}(\mu) - \mathbf{e}||}$$

$$c = \frac{W_e}{W_l}\frac{n_l}{n_e}\frac{1}{\cos\phi_l}.$$

The value of c is constant on the line. Note that since δ_l is also constant along the line that $\max(\tilde{m})$ for any parameterization occurs at the same location as $\min(\tilde{\delta}_e)$. We want to find $\min_\mu(\tilde{\delta}_e(\mu))$ on the interval $\mu \in [\mu_0, \mu_1]$ inside the view frustum. To simplify the analysis we assume a symmetric view frustum and bound $\tilde{\delta}_e(\mu)$ from below by replacing the $\cos\phi_e$ term with its smallest possible value, $\cos\theta$:

$$B(\mu) = c\frac{d_e(\mu)}{d_l(\mu)}\cos\theta \le \tilde{\delta}_e(\mu). \tag{52}$$

$\tilde{\delta}_e$ can be at most $1/\cos\theta$ times larger than B, i.e. when $\cos\phi_e = 1$. For typical fields of view, B is a fairly tight lower bound. For example, with $\theta = 30°$, $1/\cos\theta$ is only about 1.15.

We take the derivative of $B(\mu)$ to determine where it reaches its minimum value:

$$\frac{\mathrm{d}B}{\mathrm{d}\mu} = (c\cos\theta)\ \frac{\hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e})(\hat{\mathbf{z}}_l \cdot \hat{\mathbf{v}}_l)}{(\mu(\hat{\mathbf{z}}_l \cdot \hat{\mathbf{v}}_l))^2}. \tag{53}$$

The first term and the denominator of the second term are strictly positive and the $(\hat{\mathbf{z}}_l \cdot \hat{\mathbf{v}})$ term in the numerator is strictly negative. Therefore, the sign of $\mathrm{d}B/\mathrm{d}\mu$ depends only on $\hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e})$. Since $\hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e})$ is constant for all μ, the location $\mu^B_{\min}$ of $\min(B)$ must be at one of the boundaries of the interval:

$$\mu^B_{\min} = \underset{\mu\in[\mu_0,\mu_1]}{\operatorname{argmin}}\,(B(\mu)) = \begin{cases}\mu_0 & \hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e}) > 0,\\ \mu_1 & \hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e}) < 0.\end{cases} \tag{54}$$

When $\hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e}) = 0$, B is constant over the entire interval. For directional lights, the $(\mathbf{l} - \mathbf{e})$ term is replaced by the light direction $\hat{\mathbf{l}}$. Equation 54 shows that $\min(B)$ occurs on the faces where $\mathsf{L}(v)$ either enters or exits the view frustum, depending on the position of the light relative to the eye. When $\mu^B_{\min}$ is on a side face, $B(\mu^B_{\min})$ is the actual minimum for $\tilde{\delta}_e$. This can be seen from the fact that along a side face $\cos\phi_e = \cos\theta$ so $\tilde{\delta}_e = B$. Because B is never greater than $\tilde{\delta}_e$, this means that the actual minimum of $\tilde{\delta}_e$ over the interval cannot be smaller than the minimum B and must therefore be the same. Based on this observation we choose $\min(B)$ for δ_b:

$$\begin{aligned}\delta_b &= B(\mu^B_{\min})\\ &= \frac{W_e}{W_l}\frac{n_l}{n_e}\frac{d_e}{d_l}\frac{\cos\theta}{\cos\phi_l},\end{aligned} \tag{55}$$

where d_e and d_l are the values for points along the appropriate faces. At the point where $\mu^B_{\min}$ transitions from entry to exit faces or vice versa, $\min(B)$ is the same on both sets of faces. Thus the abrupt transitions in $\mu^B_{\min}$ as the light and camera move around do not cause temporal discontinuities in δ_b.

Intuitively, bounding the perspective aliasing error by selecting $\delta_l/r_t = \rho_b\delta_b/r_j$ can be thought of as ensuring that no light beam is wider than the lower bound on the width of any eye beam that it intersects.

4.4 Computing a parameterization in 3D

One approach to computing a parameterization in 3D is to simply follow the same process that we used for 2D. We start from a $\boldsymbol{\delta}(u, v)$ that is a 2×2 matrix that describes that the sample spacing distribution on the light image plane, invert $\boldsymbol{\delta}$, and integrate to get $\mathbf{F}(u, v)$. However, this approach has several complications. First, it is not clear how to compute a $\boldsymbol{\delta}$ that is a lower bound on $\tilde{\boldsymbol{\delta}}_e$. The main problem is that $\boldsymbol{\delta}$ now contains information about orientation, whereas in 2D it was simply a scalar. Second, even if we come up with an invertible $\boldsymbol{\delta}$, there is no guarantee that we can integrate it to obtain $\mathbf{F}$. The rows of $\partial\mathbf{F}/\partial\mathbf{u}$ are the gradients of multivariable functions $s(u, v)$ and $t(u, v)$. Thus the mixed partials of the row entries must be equivalent, i.e.:

$$\frac{\partial^2 s}{\partial u\partial v} = \frac{\partial^2 s}{\partial v\partial u} \quad \text{and} \quad \frac{\partial^2 t}{\partial u\partial v} = \frac{\partial^2 t}{\partial v\partial u}. \tag{56}$$

If this property does not hold for a general $\boldsymbol{\delta}^{-1}(u, v)$ then it is not the gradient of a function $\mathbf{F}(u, v)$. Finally, even if $\boldsymbol{\delta}^{-1}(u, v)$ is integrable, it is not guaranteed to be one-to-one over the entire domain covered by the shadow map.

Our approach is treat the parameterizations s and t as essentially two instances of the simpler 2D problem. We choose scalar functions $\delta_{b,s}$ and $\delta_{b,t}$ derived from Equation 55 and integrate their multiplicative inverses w.r.t. u and v, respectively, to obtain s and t. We choose our coordinate system on each face as shown in Figure 5. For the W_l term we use

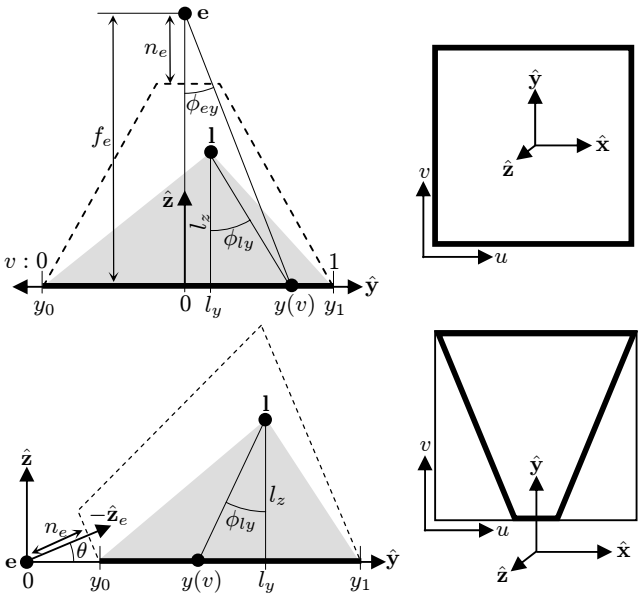

Figure 5: Frustum face coordinate systems. *For each row a side view appears on the left and a view from above the face on the right. (Top row) End face. The far face is shown here. The coordinate axes are aligned with the eye space coordinate axes with the origin at the center of the face. (Bottom row) Side face. The y axis is aligned with the center line of the face with the z axis aligned with the face normal. The origin is at the eye.*

the width of the parameterized region in the appropriate direction and for $\cos\phi_l$ we use $\cos\phi_{lx}$ or $\cos\phi_{ly}$, the angles between $\hat{\mathbf{z}}_l$ and the projection of $\hat{\mathbf{v}}_l$ in the xz or yz planes respectively. To obtain a scalar for W_e and $\cos\theta$ we assume that the fields of view are the same in both directions. If they are not, we can always choose a conservative bound $W_e = \min(W_{ex}, W_{ey})$ and $\theta = \min(\theta_x, \theta_y)$. To simplify the parameterization we also choose the light image plane to coincide with the face, thus eliminating the d_l/n_l term. We will now discuss how we compute $\delta_{b,s}$ and $\delta_{b,t}$ for each type of face and then derive the corresponding parameterizations.

End faces: We will first look at $\delta^{\text{end}}_{b,t}$. On the near face $d_e = n_e$ and on the far face $d_e = f_e$. The width of the v domain in y is the same as the width of the face $W_{ly} = (W_e d_e)/n_e$ so we can parameterize positions on the face as:

$$y(v) = (y_1 - y_0)v + v_0$$
$$y_0 = -\frac{W_e}{2}\frac{d_e}{n_e}, \quad y_1 = \frac{W_e}{2}\frac{d_e}{n_e}.$$

$\cos\phi_{ly}$ in terms of v is:

$$\cos\phi_{ly}(v) = \frac{l_z}{\sqrt{(y(v) - l_y)^2 + l_z^2}}. \tag{57}$$

Plugging these equations into Equation 55 we get:

$$\delta^{\text{end}}_{b,t}(v) = \frac{\cos\theta}{\cos\phi_{ly}(v)}. \tag{58}$$

$\delta^{\text{end}}_{b,s}$ is the same as $\delta^{\text{end}}_{b,t}$ except that quantities in y and v are exchanged for the corresponding quantities in x and u.

Side faces: The end points of the side face are related to n_e and f_e:

$$y_0 = \frac{n_e}{\cos\theta}, \quad y_1 = \frac{f_e}{\cos\theta}.$$

On a side face d_e is constant in x but increases with y. By similar triangles, the d_e/n_e term in δ_b can be expressed as:

$$\frac{d_e}{n_e} = \frac{y}{y_0}. \tag{59}$$

A side face does not cover the entire (u, v) domain. The extents of the face in x are:

$$x_0(y) = -\frac{W_e}{2}\frac{y}{y_0}, \quad x_1(y) = \frac{W_e}{2}\frac{y}{y_0}. \tag{60}$$

The width of the domain in x is the width of the face at y_1, $W_{lx} = (x_1(y_1) - x_0(y_1))$. The width in y, W_{ly}, is $(y_1 - y_0)$. We parameterize positions on the face as:

$$x(u) = (x_1(y_1) - x_0(y_1))u + x_0(y_1)$$
$$y(v) = (y_1 - y_0)v + y_0$$

Putting all these equations together yields:

$$\delta^{\text{side}}_{b,s}(u, v) = \frac{y(v)}{y_1}\frac{\cos\theta}{\cos\phi_{lx}(u)}. \tag{61}$$

$$\delta^{\text{side}}_{b,t}(v) = \frac{W_e}{(y_1 - y_0)}\frac{y(v)}{y_0}\frac{\cos\theta}{\cos\phi_{ly}(v)}. \tag{62}$$

$\delta^{\text{side}}_{b,s}$ is a function of both u and v so this is not strictly another instance of the 2D parameterization problem because the $\partial s/\partial v$ element of $\partial\mathbf{F}/\partial\mathbf{u}$ is nonzero. It is trivial to show, however, that the mixed partials of the $s(u, v)$ obtained from $\delta^{\text{side}}_{b,s}$ are equal. The parameterizations $s(u, v)$ and $t(v)$ are invertible, and thus one-to-one, because we can always find v using $t^{-1}(v)$, plug this into $s(u, v)$, and invert to find u.

Parameterizations If we let ζ_b be the indefinite integral of $1/\delta_b$, the parameterizations F_b and normalizing constants ρ_b for the three varieties of δ_b can be computed as follows:

$$F^{\text{end}}_{b,t}(v) = \frac{1}{\rho^{\text{end}}_{b,t}}\zeta^{\text{end}}_{b,t}\Big|_0^v \qquad \rho^{\text{end}}_{b,t} = \zeta^{\text{end}}_{b,t}\Big|_0^1 \tag{63}$$

$$F^{\text{side}}_{b,s}(u, v) = \frac{1}{\rho^{\text{side}}_{b,t}(v)}\zeta^{\text{side}}_{b,s}\Big|_{u_0(v)}^u \qquad \rho^{\text{side}}_{b,t}(v) = \zeta^{\text{side}}_{b,t}\Big|_{u_0(v)}^{u_1(v)} \tag{64}$$

$$F^{\text{side}}_{b,t}(v) = \frac{1}{\rho^{\text{side}}_{b,t}}\zeta^{\text{side}}_{b,t}\Big|_0^v \qquad \rho^{\text{side}}_{b,t} = \zeta^{\text{side}}_{b,t}\Big|_0^1 \tag{65}$$

$$\zeta^{\text{end}}_{b,t}(v) = C^{\text{end}}_{b,t}\ l_z \sinh^{-1}\left(\frac{y(v)-l_y}{l_z}\right) + C \tag{66}$$

$$\zeta^{\text{side}}_{b,s}(u,v) = C^{\text{side}}_{b,s}\ \frac{l_z}{y(v)} \sinh^{-1}\left(\frac{x(u)-l_x}{l_z}\right) + C \tag{67}$$

$$\zeta^{\text{side}}_{b,t}(v) = -C^{\text{side}}_{b,t}\ \frac{l_z}{\sqrt{l_y^2+l_z^2}} \log\left(\frac{2}{y(v)}\left(\frac{l_y(l_y-y(v))+l_z^2}{l_z\sqrt{l_y^2+l_z^2}} + \frac{1}{\cos\phi_{ly}(v)}\right)\right) + C. \tag{68}$$

$$C^{\text{end}}_{b,t} = \frac{1}{(y_1-y_0)\cos\theta} = \frac{n_e}{W_e d_e \cos\theta}$$

$$C^{\text{side}}_{b,s} = \frac{1}{(x_1(y_1)-x_0(y_1))\cos\theta} = \frac{n_e}{W_e f_e \cos\theta}$$

$$C^{\text{side}}_{b,t} = \frac{y_0}{W_e \cos\theta} = \frac{n_e}{W_e \cos^2\theta}$$

Note that for $F^{\text{side}}_{b,s}$ we integrate over $u \in [u_0(v), u_1(v)]$ corresponding to the part of the domain covered by the face. $u_0(v)$ and $u_1(v)$ can be found by solving $x(u) = x_0(y(v))$ and $x(u) = x_1(y(v))$ for u:

$$u_0(v) = \frac{y_1 - y(v)}{2y_1}, \quad u_1(v) = \frac{y_1 + y(v)}{2y_1}. \tag{69}$$

Unfortunately, the parameterizations based on δ_b are too complex for practical use. In the next section, we will examine approximations to δ_b that have simpler parameterizations. (Appendix A includes further analysis of the parameterizations based on δ_b).

5 Deriving the LogPSM parameterization

In this section, we seek spacing functions that produce error that is nearly as low as δ_b, but that have simpler parameterizations. We first discuss the global error metrics that we use to compare different parameterizations. We then analyze several parameterizations to identify those with error on the same order as the parameterizations for $\delta^{\text{end}}_{b,t}$, $\delta^{\text{side}}_{b,s}$, and $\delta^{\text{side}}_{b,t}$. We use these simpler parameterizations to formulate our LogPSM parameterization.

5.1 Global error measures

We can compare parameterizations in 2D using the critical resolution factor R (Equation 47) as an error metric. R is directly related to the maximum perspective aliasing error over the entire frustum. In 3D we combine the critical resolution factors for s and t:

$$S = R_s \times R_t. \tag{70}$$

We call S the *storage factor* because it represents the overall size in texels of a critical resolution shadow map relative to the size of the image in pixels. The use of the critical resolution and storage factor as error measures was originally introduced by Lloyd et al. [2006]. Note that $\delta^{\text{side}}_{b,s}$ is a function of u and v, so R^{side}_s will be:

$$R^{\text{side}}_s = \max_{(u,v)\in\mathcal{F}} \left(\frac{\delta_l(u,v)}{\delta^{\text{side}}_b(u,v)}\right), \tag{71}$$

Param.	$\mathsf{R}^{\text{end}}_s$	$\mathsf{R}^{\text{side}}_s$	$\mathsf{R}^{\text{side}}_t$	S^{side}
Bound	$O(1)$	$O(1)$	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$
Uniform	$O(1)$	$O\left(\frac{f_e}{n_e}\right)$	$O\left(\frac{f_e}{n_e}\right)$	$O\left(\left(\frac{f_e}{n_e}\right)^2\right)$
Perspective	–	$O(1)$	$O\left(\sqrt{\frac{f_e}{n_e}}\right)$	$O\left(\frac{f_e}{n_e}\right)$
Logarithmic	–	–	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$	$O\left(\log\left(\frac{f_e}{n_e}\right)\right)$

Table 2: Maximum error *This table shows measures of the maximum error over all light directions* R *and* S *for our error bound parameterization* F_b *and several simpler ones. The perspective and logarithmic parameterizations have error that is on the same order as* $F_{b,s}$ *and* $F_{b,t}$ *for side faces. These parameterizations form the basis of the LogPSM parameterization.*

where $\mathcal{F}$ is the set of points covered by the face. It is also useful to define measures of maximum perspective error over all possible light positions Ω:

$$\mathsf{R} = \max_{\Omega} R \tag{72}$$

$$\mathsf{S} = \max_{\Omega} S\,. \tag{73}$$

R and S can be evaluated simply by computing R and S for a light at infinity directly above the face, in which case the $\cos\phi_l$ factor of δ_b is 1 over the entire face and R and S reach their maximum values.

5.2 Maximum error of various parameterizations

Table 2 summarizes the error of four different parameterizations for end faces and both directions on a side face (the values for the table are derived in Appendix A). The first is the error bound parameterization F_b derived in the last section. The next three are approximations for δ_b:

- **Uniform.** The simplest parameterization is the uniform parameterization corresponding to a standard shadow map fit to the view frustum:

$$\begin{bmatrix} s \\ t \end{bmatrix} = \mathbf{F}_{un}(u,v) = \begin{bmatrix} u \\ v \end{bmatrix}. \tag{74}$$

- **Perspective.** The next parameterization is a perspective projection along the y axis of the side face:

$$\begin{bmatrix} s \\ t \end{bmatrix} = \mathbf{F}_{p}(u,v) = \begin{bmatrix} \frac{p_0 x(u) + p_1 (y(v)+a)}{y(v)+a} \\ \frac{p_2 (y(v)+a) + p_3}{y(v)+a} \end{bmatrix} \tag{75}$$

$$p_0 = \frac{(y_1+a)}{W_e(y_1/y_0)}$$

$$p_1 = \frac{1}{2}$$

$$p_2 = \frac{y_1+a}{y_1-y_0}$$

$$p_3 = -p_2(y_0+a), \tag{76}$$

  where a is a free parameter that translates the center of projection to a position of $y = -a$. $a = 0$ yields the standard perspective projection. As $a \to \infty$, the parameterization degenerates to $\mathbf{F}_{un}$. The perspective parameterization is used by existing shadow map warping algorithms [Stamminger and Drettakis 2002; Wimmer et al. 2004; Martin and Tan 2004]. These algorithms differ essentially in the way they choose a. For

the values in Table 2 we have computed the optimal parameter with respect the given error metric.

- **Logarithmic.** The last parameterization is logarithmic:

$$t = F_{\log}(v) = \frac{\log\left(\frac{y(v)+a}{y_0+a}\right)}{\log\left(\frac{y_1+a}{y_0+a}\right)}. \tag{77}$$

A logarithmic parameterization has been suggested as a good fit for perspective aliasing [Wimmer et al. 2004; Zhang et al. 2006a; Lloyd et al. 2006]. It produces a spacing distribution that increases linearly in v and can thus match the $y(v)$ term in $\delta^{\text{side}}_{b,t}$. This is a generalized version of a logarithmic parameterization with a free parameter a similar to that of $\mathbf{F}_p$.

The values in Table 2 are expressed in terms of the ratio of the far to near plane distances of the view frustum. It is this term by which the error is predominantly determined. There is also some dependence on the field of view, but the typical range of values for the field of view used in interactive applications is fairly limited. Further analysis of these parameterizations is found in Appendix A.

We are looking for approximate parameterizations that produce error that is at least on the same order as that of F_b. From Table 2 we see that suitable approximations to δ_b can be obtained by using a uniform parameterization for the end faces, a perspective projection for s on a side face, and a logarithmic parameterization for t. Our LogPSM parameterization combines a logarithmic transformation with a perspective projection to get good bounds for both s and t on a side face.

5.3 Logarithmic + perspective parameterization

Multiplying by a constant and adding another we can transform $F_{p,t}$ into $(y_0+a)/(y(v)+a)$. We then use the fact that $\log(c/d) = -\log(d/c)$ to then obtain $F_{\log}$. Our logarithmic perspective parameterization is given by:

$$\begin{bmatrix} s \\ t \end{bmatrix} = \mathbf{F}_{lp}(u,v) = \begin{bmatrix} F_{p,s}(u,v) \\ c_0 \log(c_1 F_{p,t}(u,v) + c_2) \end{bmatrix} \tag{78}$$

$$c_0 = \frac{-1}{\log\left(\frac{y_1+a}{y_0+a}\right)}$$

$$c_1 = \frac{y_0+a}{p_3} = -\frac{y_1-y_0}{y_1+a}$$

$$c_2 = -(y_0+a)\frac{p_2}{p_3} = 1$$

With $a = 0$ this parameterization provides the same good error bounds in the s and t directions as the perspective and logarithmic parameterizations, respectively. See Appendix A for more details.

6 LogPSM Algorithms

In this section we outline several algorithms that use the LogPSM parameterization. These algorithms are basically extensions of LiSPSMs [Wimmer et al. 2004], cascaded shadow maps [Engel 2007; Zhang et al. 2006a], and perspective-warped cube maps [Kozlov 2004]. The first algorithm uses a single shadow map to cover the entire view frustum. The second algorithm partitions the frustum along its z-axis into smaller sub-frusta and applies a single shadow map to each one. The third algorithm uses a separate shadow maps for partitions corresponding to the view frustum faces. The first two algorithms are best suited for directional lights or spot lights, while the third algorithm can support both directional and omnidirectional point lights. Each algorithm consists of several steps:

1. Compute the parameterizations for the shadow maps corresponding to each partition. The partitions may consist of the entire view frustum, z-partitioned sub-frusta, or face partitions.
2. Error-based allocation of texture resolution.
3. Rendering the shadow maps.
4. Rendering the image with multiple shadow maps.

We will discuss each of these steps in detail. We will then describe modifications to handle shearing artifacts that can sometimes arise with the face partitioning algorithm.

6.1 Parameterizing partitions.

The perspective portion of the LogPSM parameterization determines the shape of the light frustum used to render the shadow map. To maximize the use of the available depth and shadow map resolution, the light frustum should be fit as tightly as possible to the relevant portions of the scene. If depth clamping is available, the near and far planes can be set to bound only the geometry that is visible to the eye. Depth information is only needed to prevent false self-shadowing on visible occluders. For unseen occluders, a single value indicating occlusion is sufficient. With depth clamping on, the depth of occluders between the light position and the near plane will be set to 0. If the near plane is set such that visible surfaces will always have depth greater than 0, the shadow computation will be correct. If depth clamping is not available, the light frustum must also include all potential occluders. When the light itself lies inside the view frustum, surfaces between the light and the near plane of the light frustum will not be shadowed. Therefore, the near plane should be chosen close enough to minimize the unshadowed region, but far enough away to preserve depth resolution. Unclamped floating point depth buffers, such as those supported by the recent *NV_depth_buffer_float* extension, could eliminate the unshadowed region.

To obtain tight bounds on the visible geometry we fit each light frustum to the convex polyhedron P of its partition. If depth clamping is not available, each P can be expanded to include potential occluders by taking the convex hull of P and the light position $\mathbf{l}$. Each P can also be intersected with the scene bounds to obtain tighter bounds.

6.1.1 Entire view frustum

We use the LiSPSM algorithm [Wimmer et al. 2004] as the basis for applying the LogPSM parameterization to a single shadow map that covers the entire view frustum. LiSPSMs degenerate to a uniform parameterization in order to avoid excessive error as the angle γ between the light and view directions approaches 0° or 180°. A uniform parameterization is produced when near plane distance n' of the perspective

parameterization diverges to ∞. (For an overhead directional light $n' = a + n_e$.) To accomplish this, the LiSPSM algorithm modulates n' with $1/\sin\gamma$:

$$n' = \frac{n_e + \sqrt{n_e f_e}}{\sin\gamma}.$$

A LogPSM could use the same strategy, exchanging n_e for $n_e + \sqrt{n_e f_e}$:

$$n' = \frac{n_e}{\sin\gamma}.$$

Unfortunately, for some γ this can result in error that is worse than the error of standard shadow maps. Essentially, the problem is that $\sin\gamma$ does not go to 0 fast enough for $\gamma \in [0°, \theta]$ and $\gamma \in [180°, 180° - \theta]$. The problem is worse for LogPSMs than for LiSPSMs because n' has farther to go to get to ∞. Instead, we use the following function to compute n' for LogPSMs:

$$\eta = \begin{cases} 0 & \gamma < \gamma_a \\ -1 + (\eta_b + 1)\frac{\gamma - \gamma_a}{\gamma_b - \gamma_a} & \gamma_a < \gamma < \gamma_b \\ \eta_b + (\eta_c - \eta_b)\sin\left(90\frac{\gamma - \gamma_b}{\gamma_c - \gamma_b}\right) & \gamma_b < \gamma < \gamma_c \\ \eta_c & \gamma < \gamma_c \end{cases} \quad (79)$$

$$\gamma_a = \frac{\theta}{2}, \quad \gamma_b = \theta, \quad \gamma_c = \theta + 0.4(90 - \theta)$$

$$\eta_b = -0.2, \quad \eta_c = 1.$$

$\eta \in [-1, 1]$ is a parameterization of n' that corresponds to $n' = \infty$ at $\eta = -1$, $n' = n_e + \sqrt{n_e f_e}$ at $\eta = 0$, and $n' = n_e$ at $\eta = 1$. The shaping function for η in Equation 79 smoothly rises from -1 to η_b over the interval $[\gamma_a, \gamma_b]$ and then continues to rise until it smoothly transitions to η_c at $\gamma = \gamma_c$. This function provides a continuous transition to $n' = \infty$ but with more control than the simple $\sin\gamma$ function. We convert η to n' using the equations presented in [Lloyd et al. 2006]. Note that this shaping function can also be used to improve the LiSPSM algorithm by using $\gamma_a = \theta/3$ and $\eta_c = 0$. A more complete analysis of this function and our choice of parameters is found in Appendix B.

The perspective portion of the LogPSM parameterization is encoded in a matrix $\mathbf{M}_s$ that is used during shadow map rendering. This matrix is computed as with LiSPSMs, but with our modified n'. Note that the light space z-axis used in the LiSPSM algorithm corresponds to our y-axis.

6.1.2 z-partitioning

The idea of z-partitioning schemes is to subdivide the view frustum along its z-axis so that tighter fitting shadow maps may be applied to each sub-frusta. The main difference between the various z-partitioning algorithms is where subdivisions are made. The maximum error over all partitions is minimized when the split points are chosen such that the far to near plane distance ratio of each partition is the same:

$$n_i = n_e \left(\frac{f_e}{n_e}\right)^{(i-1)/k} \quad (80)$$

$$f_i = n_{i+1} = n_e \left(\frac{f_e}{n_e}\right)^{i/k} \quad (81)$$

$$\frac{f_i}{n_i} = \left(\frac{f_e}{n_e}\right)^{1/k}. \quad (82)$$

where k is the number of partitions and $i \in \{1, 2, ..., k\}$. We use this partitioning scheme in this paper. Zhang et al. [2006a] use a combination of this scheme with a uniform partitioning. This algorithm favors regions further from the viewer. Once the z-partitions have been computed, applying the LogPSM parameterization proceeds just as in the single shadow map case.

6.1.3 Face partitioning

Face partitioning is motivated by the analysis in Section 5, which showed that the function that bounds the maximum perspective aliasing error changes between regions corresponding to the frustum faces. We use a uniform parameterization for the end faces of the view frustum and the LogPSM parameterization for the side faces. The uniform and the perspective portion of parameterizations can be handled using the PSM cube map algorithm of Kozlov [2004]. The main idea is to parameterize the faces of the view frustum in the post-perspective space of the camera. First, the light $\mathbf{l}$ is transformed to $\mathbf{l}'$ by the camera matrix $\mathbf{M}_c$. Under this transformation, the view frustum is mapped to the unit cube. Second, a projection matrix for the light $\mathbf{M}_l$ is fit to each back face of the unit cube with the light frustum's near plane oriented parallel to the face. $\mathbf{M}_l$ is a simple orthographic projection if $\mathbf{l}'$ is a directional light (homogeneous coordinate l'_w is 0) or an off-centered perspective projection if $\mathbf{l}'$ is a point light. If l'_w is negative, the depth order relative to the light becomes inverted. To restore the proper depth order, the row of $\mathbf{M}_l$ corresponding to the z-axis should be scaled by -1 (see Figure 6). The inversion occurs whenever the face selection criterion in Equation 54 selects the front faces ($\hat{\mathbf{z}}_e \cdot (\mathbf{l} - \mathbf{e}) > 0$). Thus parameterizing back faces in post-perspective space always selects the appropriate faces. The partitions polyhedra can be computed by taking the convex hull of the back faces of the unit cube and the light and intersecting them with the unit cube. The final matrix used to render each shadow map is obtained by concatenating the camera and light projection matrices, $\mathbf{M}_s = \mathbf{M}_l\mathbf{M}_c$.

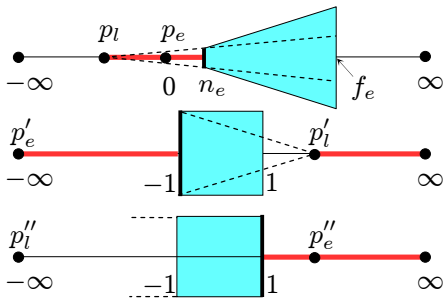

Figure 6: Post-perspective parameterization of frustum faces. (Top) *With the light p_l behind the eye p_e we parameterize the front face. The red points along the line contain potential occluders and must be included in the shadow map.* (Middle) *In post-perspective space the frustum becomes a unit cube, the eye goes to $-\infty$ and the light wraps around ∞. The front face is now a back face.* (Bottom) *After applying the light's perspective transform, p''_l is at $-\infty$ and the projection becomes orthographic. The inverted depth ordering can be corrected by leaving the light at $-\infty$ and scaling by -1 along the view direction.*

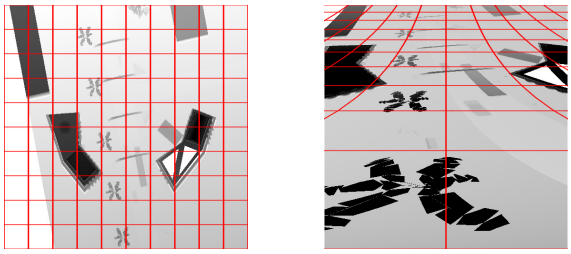

*Figure 7: **Standard and logarithmic shadow maps.** A uniform grid is superimposed on the shadow map to show the warping from the logarithmic parameterization. Note that straight lines become curved*

6.2 Error-based allocation of texture resolution.

We render the shadows in the image in a single pass, storing all the shadow maps in memory at once. We used a fixed texture resolution budget that must be distributed somehow among the shadow maps. The simplest approach would be to allocate a fixed amount of resolution in each direction for each shadow map. One problem with this strategy is that as the light position varies relative to the eye the maximum error can vary dramatically between partitions and shadow map directions (see Figure 9a). To keep the overall error as low as possible, we distribute the error evenly between the partitions and between the s and t directions by adjusting the resolution as described by Lloyd et al. [2006]. The idea is to first compute R_s, R_t, and S for each partition. Each partition P_i then receives a fraction $\sigma_i = S_i / \Sigma_{j=1}^{k} S_j$ of the total texel budget T. The texel allocation for a partition is then divided between the s and t directions in the shadow map in proportion to R_s and R_t:

$$r_s = \sqrt{\frac{R_s}{R_t}\sigma T}, \qquad r_t = \sqrt{\frac{R_t}{R_s}\sigma T}. \tag{83}$$

The effect of this resolution redistribution is shown in Figure 9b.

We compute R_s and R_t with Equation 47 treating each direction independently, as we did in Section 4. For R_s we use $W_l = W_{lx}$ and $\cos\phi_l = \cos\phi_{lx}$, and for R_t we use $W_l = W_{ly}$ and $\cos\phi_l = \cos\phi_{ly}$. The only term in δ_b that varies for a directional light is d_e because $n_l/d_l \to 1$ and $\cos\phi_l$ is constant. $\mathrm{d}F_{lp,s}/\mathrm{d}u$ and $\mathrm{d}F_{lp,t}/\mathrm{d}v$ are monotonic, therefore the error distribution is monotonic over each face of the view frustum. This means that we can compute R_s and R_t by evaluating δ_l/δ_b only at the vertices of the view frustum faces and taking the maximum. For a point light, computing the maximum of δ_l/δ_b over a face can be more involved. As an approximation we also evaluate the error at the point on each face that is closest to the light.

Current GPUs drivers are not necessarily optimized to handle the scenario of a texture changing resolution with each render. For the experimental results presented in this paper we allocated the shadow maps in a single texture atlas large enough to accommodate the changing shadow map dimensions.

6.3 Rendering the shadow maps.

Each shadow map is rendered in a separate pass. In order to maximize rendering performance for each pass, objects should only be rendered into the shadow maps in which they actually lie. We call this *partition culling*. Partition culling can be accomplished by testing an object against the planes through the light and the partition silhouette edges using straightforward extensions of the existing view frustum culling techniques typically employed in graphics applications. For simple scenes, partition culling is of only small benefit and may not be worth the overhead, but for geometrically complex scenes this can be an important optimization. GPUs that support DirectX 10 features can render to multiple render targets in a single pass, but if the vertex data already resides on the GPU, this does little to reduce the overhead of multipass rendering. So partition culling can still be an important optimization here as well.

```
void logRasterize(
  varying float3 stzPos,
  varying float3 edgeEq0,
  varying float3 edgeEq1,
  varying float3 edgeEq2,
  varying float3 depthEq,
  uniform float  d[3],
  out float depth : DEPTH )
{
  // invert transform
  float3 p = stzPos;
  p.y = d[0]*exp( d[1]*p.y )+d[2];

  // test against edges
  if( dot( p, edgeEq0 ) < 0  ||
      dot( p, edgeEq1 ) < 0  ||
      dot( p, edgeEq2 ) < 0    )
    discard;

  // compute depth
  depth = dot(p, depthEq);
}
```

Figure 8: CG code used to simulate logarithmic rasterization in a fragment program.

Logarithmic rasterization. The LogPSM parameterization requires logarithmic rasterization. Logarithmic rasterization causes planar primitives to become curved (see Figure 7). Current GPUs only support projective transformations. The logarithmic transformation could be computed on vertices in a vertex program, but when the f_e/n_e ratio is high this would require a very fine tessellation of the scene to avoid error. We currently simulate logarithmic rasterization by performing brute-force rasterization in a fragment program. We first render the scene using $\mathbf{M}_s$ with the viewport set to the range $[0,0] \times [1,1]$ and readback the clipped triangles using the OpenGL feedback mechanism. This gives us the triangles transformed by $\mathbf{F}_p(u,v)$. We compute the edge equations and depth interpolation equation for each triangle. We then transform the vertices of each triangle by $\mathbf{F}_{lp}$ and render a bounding quad. We use the fragment program in Figure 8 to invert $F_{lp,t}$ in order to undo the logarithmic warping on the position of each fragment, compare against the linear triangle edge equations, and discard fragments that fall outside the triangle. The equation for the inverse of $F_{lp,t}(v)$ is given by:

$$v = d_0 e^{d_1 t} + d_2 \tag{84}$$
$$d_0 = \frac{1}{c_1(y_1 - y_0)}, \qquad d_1 = \frac{1}{c_0}, \qquad d_2 = -\frac{c_1 y_0 + c_2}{c_1(y_1 - y_0)}.$$

With our unoptimized simulator we observe frame rates of 2–3 fps on a PC with a GeForce 6800GT graphics card and a

2.8 GHz processor. Most of that time is spent in computing the bounding quads. GPUs that support geometry shaders can probably be used to create the bounding quads for the warped triangles much more efficiently, but we have not yet implemented this. Even with a geometry shader, however, logarithmic rasterization would likely be considerably slower than linear shadow map rasterization. GPUs have a number of optimizations which are disabled when a fragment program outputs depth information, such as z-culling and higher rasterization rates for depth-only rendering.

Logarithmic rasterization can be supported in graphics hardware [Lloyd et al. 2007] to allow LogPSMs to be rendered the same speeds as other algorithms that use a similar number of shadow maps [Kozlov 2004; Zhang et al. 2006a]. The LogPSM parameterization is especially practical for hardware implementation because the standard graphics pipeline can be used for the perspective portion of the parameterization. Only the rasterizer needs to be modified to handle the logarithmic part. For the same error as other algorithms, LogPSMs can require less memory bandwidth and storage. This is important because shadow map rendering is often bandwidth limited and high resolution shadow maps increase contention for limited GPU memory resources.

6.4 Rendering the image

We render the image using all the shadow maps in a single pass. The entire frustum algorithm is simplest because it uses only one shadow map. We compute texture coordinates for the perspective part of the parameterization at the vertices using the transformation $\mathbf{M}_n\mathbf{M}_s\mathbf{p}$ where $\mathbf{M}_n$ maps the $[-1,-1]\times[1,1]$ range of $\mathbf{M}_s$ to $[0,0]\times[1,1]$ and $\mathbf{p}$ is the world position. We linearly interpolate these texture coordinates over the triangle. In the fragment program we perform the perspective divide on the texture coordinates and apply the logarithmic transformation in Equation 78 to the y coordinate to get the final texture coordinates for the shadow map lookup.

z-partitioning introduces additional shadow maps. In the fragment program we must determine in which partition the fragment lies. This can be done efficiently with a conditional and a dot product [Engel 2007]:

```
float4 zGreater = (n1_4 < eyePos.z);
float  partitionIndex = dot(zGreater, 1.0f);
```

The constant `n1_4` contains the near plane distances of the first four z-partitions (n_1, n_2, n_3, n_4). This method is easily extended to handle more partitions. If a sufficient number of interpolators are available, we can interpolate a set of texture coordinates for each shadow map and use the partition index to select the appropriate set. Otherwise we use the index to select the appropriate element of an array of the matrices $\mathbf{M}_s$ for each partition and perform the matrix multiplication on the world position in the fragment program. Once we have the right set of texture coordinates, the computation proceeds just as in the single shadow map case.

For face partitioning we compute the partition index using a small cube map. In each face of the cube map we store the index of the partition corresponding to that face. We perform the cube map lookup using the method described by Kozlov [2004]. We do not store the shadow maps directly in a cube map because cube maps do not currently support non-square textures of differing resolution or the necessary logarithmic transformation. We also store a flag in the cube map indicating whether the face is a side face or an end face so we know whether or not to apply the logarithmic transformation.

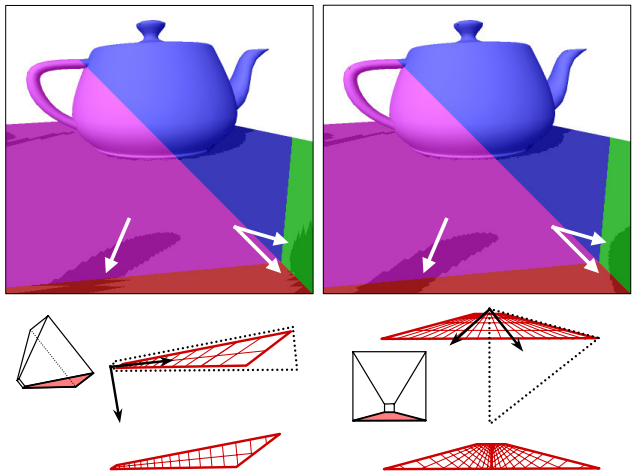

Figure 10: Handling shear artifacts. (Top-left) *Shear artifacts resulting from face partitioning. We use a low-resolution shadow map to make them easier to see. Regions are colored according to the face partition to which they belong.* (Top-right) *Shear artifacts removed by coordinate frame adjustment.* (Bottom) *View frustum as seen from the light in cases in which shearing occurs. To handle the first case, we realign the projection axis with the bisector of the side edges of the face make the other axis orthogonal in the light view. For the second case, we first split the face along the bisector into two sub-faces and apply the coordinate frame adjustment to each of the sub-faces. The corrected faces are shown below.*

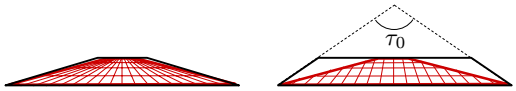

Figure 11: τ-Limiting. *(Left) a face partition with excessive shearing caused by a large angle τ between the edges of the trapezoidal region to which to parameterization is applied (black). (Right) Limiting τ to some constant τ_0 reduces the shearing at the expensive of error at the narrow end of the face partition.*

6.5 Handling shear

The PSM cube map algorithm that we extend for our face partitioning algorithm fixes the parameterization to the face. For some light positions, the cross-section of the light beams become overly sheared which can lead to disturbing artifacts (see Figure 10). With any shadow map algorithm some shearing can occur on surfaces that are nearly parallel to the light. However, this shearing is usually less noticeable because diffuse shading reduces the intensity difference between shadowed and unshadowed parts of these surfaces. Warping algorithms fit a rectangular shadow map to the trapezoidal view frustum introducing some shearing on surfaces that are directly facing the light. This shearing is more pronounced with face partitioning because the trapezoid regions as seen from the light can become extremely sheared and flattened for some light positions. In addition, the shearing in one partition may not be consistent with that of adjacent partitions and is therefore more noticeable. Because our error metrics do not take this shearing into account, redistributing the resolution according to error may also make this shearing more noticeable.

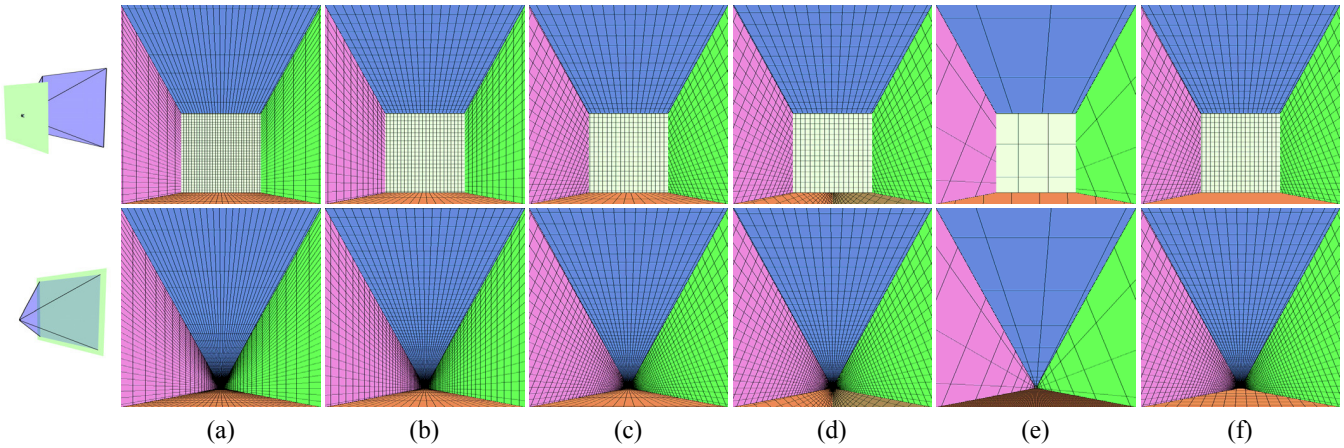

Figure 9: The effects of resolution redistribution and shear handling. *The top row shows the texel grid of the shadow map projected onto a plane near the viewer that is oriented perpendicular to the view direction. The bottom row uses a plane at the other end of the view frustum. (a) The same resolution is used in s and t for all partitions. The error is distributed unequally. (b) Resolution is distributed according to maximum error resulting in a more uniform parameterization. (c) Coordinate frame adjustment alleviates excessive shearing in the upper corners of the left and right face partitions, but does nothing for the bottom one. (d) Splitting the bottom face and adjusting the coordinate frame alleviates shearing. (e) Limiting the field of view of the parameterization applied to the bottom face results in high error. Resolution redistribution leaves little for the other partitions. (f) Using the original resolution allocation leads to high error close to viewer but acceptable results further away.*

One solution is to unbind the parameterization from the face. Consider the view of a side face from a directional light. The shearing of the light beam cross-sections can be minimized by fitting the parameterization to the symmetric trapezoid that bounds the face. We align the midline of the trapezoid with the bisector of the two side edges of the face (see Figure 10). Projecting this trapezoid into the plane of the face, we can see that this approach effectively shears and rotates the coordinate frame used for the parameterization. The resulting projection is no longer a tight fit, but greatly reduces the shear artifacts. However, when the angle between the two edges is high, the coordinate frame adjustment may provide little improvement (see the red face in Figure 9c). Therefore we first split the face along its bisector and then apply the coordinate frame adjustment to each half. We split a face when the angle between the side edges τ exceeds a specified threshold τ_0. To avoid a sudden pop when a face is split, we specify another threshold $\tau_1 > \tau_0$ and "ease in" to the new coordinate frame over the interval $[\tau_0, \tau_1]$. Choosing $\tau_0 > 90°$ ensures that no face will be split more than once and that no more than two faces will be split at the same time.

The image rendering pass needs to be modified slightly to handle split faces. In the cube map we store the indices of the shadow maps for both halves of the face. The second index goes unused for unsplit faces. We compute the equations of the plane containing the light and split line and pass these to the fragment program. We test the fragment world position against the plane equation of the corresponding face to determine which index to use. We adopt the convention that the first index corresponds to the negative side of the plane. For faces without a split we use the plane equation $(0, 0, 0, -1)$ which is guaranteed to always give a negative result. Once the appropriate index has been computed, the calculations proceed in the same way as before.

Another possibility for handling shear is to simply place a limit τ_0 on τ (see Figure 11). This approach does not require additional shadow maps. However, it leads to large errors on the parts of the face close to the near plane. Naive resolution redistribution allocates most of the resolution to the problem face. The maximum error on the face is equalized with the other faces, but the error over the entire view frustum goes up (Figure 9e). We could also use the resolution that would have been allocated to the face had we not changed the parameterization. For surfaces near the view the error may be extremely high at the narrow end of the face partition, but acceptable for surfaces farther away (Figure 9f). Depending on the application, this may not be a problem, especially since the high error situations occur for face partitions that cover a very small part of the view frustum. A compromise might be to use a resolution for the face that is some blend of the two extremes.

For both of these approaches we find it easier to work in light space rather than the post-perspective space of the camera. We can compute the partitions in post-perspective space and transform them back into light space. Then we parameterize the partitions on a light image plane that is perpendicular to the light direction. For a point light there is no one light direction. Due to this and other complications we currently only handle shear for directional lights.

7 Results and Analysis

In this section, we present empirical results for LogPSMs obtained using our simulator for logarithmic rasterization. We also perform several comparisons between different shadow map algorithms. We use the following abbreviations to classify the different algorithms:

- P: Perspective warping. Unless indicated otherwise, we use the LiSPSM parameter and our new shaping function for the warping parameter.
- Po: Perspective warping with $1/\sin\gamma$ falloff.
- LogP: Logarithmic + perspective warping.
- ZPk: z-partitioning using k partitions.
- FP: Face partitioning.

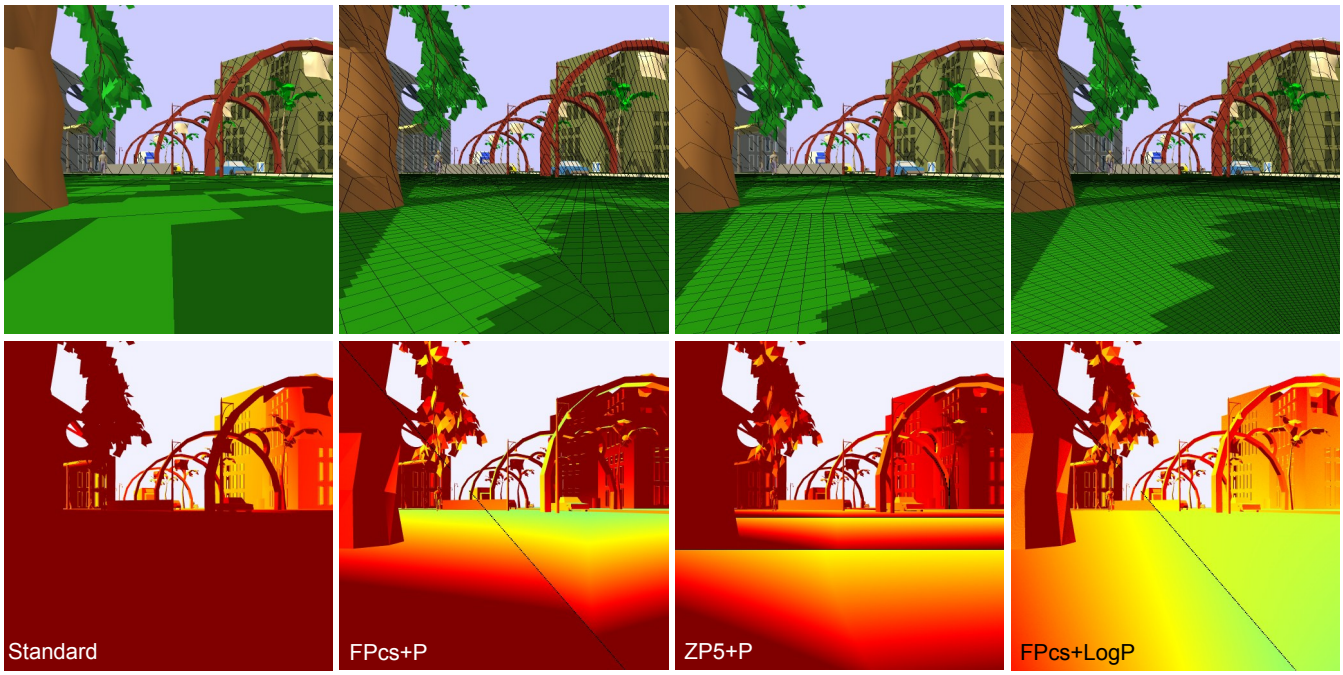

Figure 12: Comparison of various algorithms. *The viewer is positioned below a tree in a town scene. (*Top row*) Grid lines for every 5 texels projected on the scene. (*Bottom row*) Color coding of aliasing error in terms of projected shadow map texel area in pixels. FPcs+P is a combination of face partitioning and perspective warping. ZP5+P uses 5 z-partitions combined with perspective warping. FPcs+LogP is a combination of face partitioning with a logarithmic perspective warping (see Section 7 for more detail on the abbreviations). The face partitioning algorithms use only 3 shadow maps for this view. Pixels are black at partition boundaries where derivatives are not well-defined. Both the image and* total *shadow map resolutions are* 512×512*. The LogPSM produces lower, more evenly distributed error. (*$f/n = 1000$*,* $\theta = 30°$*).*

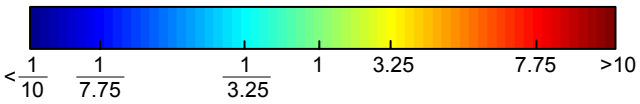

Figure 13: Color mapping for error comparison images.

- FPc: Face partitioning with coordinate frame adjustment to handle shearing.

- FPcs: Face partitioning with coordinate frame adjustment and face splitting.

Partitioning schemes can be combined with warping, e.g. ZP5+P stands for z-partitioning with 5 partitions and perspective warping, and FPc+LogP is face partitioning with coordinate frame adjustment and the logarithmic+perspective warping. When ZP5 appears alone, a uniform parameterization is used.

Showing the quality of one shadow map algorithm relative to another from images alone is often difficult. The regions of maximum error can differ between algorithms. To see the error, surfaces must pass through these regions. Moreover shadow edges must also be present in these regions. In order to more easily visualize the aliasing error, we project texel grid lines from the shadow map onto the scene. In addition, we generate color coded images of aliasing error over the whole image using the maximum extent in pixels of the projected texel in the image as the aliasing metric. We measure the maximum extent as the maximum of the diagonals so as to take into account the error in both directions.

$$m = \max\left(|\mathbf{ds} + \mathbf{dt}|, |\mathbf{ds} - \mathbf{dt}|\right) \tag{85}$$

$$\mathbf{ds} = \left(\frac{r_i}{r_s}\frac{\partial i}{\partial s}, \frac{r_j}{r_s}\frac{\partial j}{\partial s}\right), \quad \mathbf{dt} = \left(\frac{r_i}{r_t}\frac{\partial i}{\partial t}, \frac{r_j}{r_t}\frac{\partial j}{\partial t}\right). \tag{86}$$

The color mapping used for the comparison images is shown in Figure 13.

Figure 12 shows a comparison between various shadow map algorithms. The aliasing is extremely high near the viewer for the standard shadow map, but much better in the distance. The FPc+P algorithm is comparable to Kozlov's perspective warped cubemap algorithm [2004] except that the LiSPSM parameter is used for warping instead of the PSM parameter. This gives a more uniform distribution of error. FPcs+LogP has lower error than FPcs+P due to the better parameterization. ZP5+P is similar to cascaded shadow maps [Engel 2007], but adds warping for further error reductions. ZP5+P always renders 5 shadow maps while FPcs+LogP can render anywhere from 1 to 7. For this view, FPcs+LogP renders only 3. FPcs+LogP has the most even distribution of error.

Figure 14 shows a dueling frustum situation which is especially difficult for single shadow map algorithms to handle. Here FPc+LogP produces less error than ZP5+P. The portion of the image around the light direction is over sampled for surfaces far from the viewer.

Figure 16 is a comparison using a single shadow map. The light is nearly in the optimal position for both P and LogP. When the light is behind or in front of the viewer, both of these algorithms degenerate to a standard shadow map.

Figure 1 shows FP+LogP used with point lights. We compare the algorithm against Kozlov's perspective warped cube

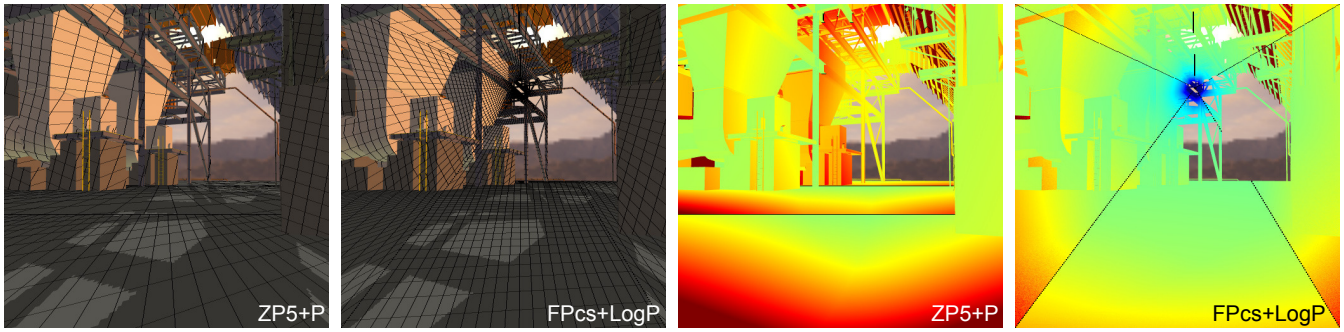

Figure 14: Comparison in a power plant model. *The light is placed almost directly in front of the viewer. This is the dueling frustum case that is difficult for single shadow map algorithms to handle. Both FP and ZP algorithms can handle this situation well, though FPcs+LogP produces less error than ZP5+P. The image resolution is* 512×512 *and the total shadow map resolution is* 1024×1024*.* $(f/n = 1000,\ \theta = 30^\circ)$

Algorithm	Min S	Max S	Max S/Min S	Mean	Rel. mean	Mean # SMs
Standard	8.65×10^5	2.00×10^6	2.31	1.32×10^6	1.21×10^5	1
P	865	3.65×10^6	4.22×10^3	2.53×10^5	2.32×10^4	1
Po	865	1.99×10^6	2.30×10^3	1.80×10^5	1.64×10^4	1
Log	5.98	1.99×10^6	3.33×10^5	1.88×10^5	1.71×10^4	1
FP+P	865	2000	2.31	1490	136	3.6
FPc+P	865	3170	3.66	1860	170	3.6
FPcs+P	865	3630	4.19	1900	174	3.8
ZP5	51.4	158	3.07	94.1	8.60	5
ZP5+P	12.9	159	26.4	46.7	4.93	5
ZP5+Log	5.98	158	26.4	46.7	4.26	5
ZP7	27.4	100	3.67	56.6	5.17	7
ZP7+P	10.2	100	9.84	42.1	3.85	7
ZP7+Log	5.98	100	16.8	38.5	3.51	7
FP+Log	5.98	14.8	2.47	10.9	1	3.6
FPc+Log	5.98	26.3	4.4	14	1.28	3.6
FPcs+Log	5.98	31.7	5.3	14.3	1.31	3.8

Table 3: Storage factor over all light directions. *Since the storage factor is greatest as the light moves toward infinity, we used a directional light in order to obtain an upper bound on the storage factor. This table summarizes statistics for various combinations of perspective warping (P), logarithmic+perspective warping (LogP), face partitioning (FP), and z-partitioning (ZP). The second to last column shows the mean storage factor relative to FP+LogP. The last column shows the mean number of shadow maps used. Over all light directions LogPSMs have the lowest minimum storage factor. FP+Log and its variations also have the lowest maximum, and mean storage factor. The values in the table do not include the* $1/\cos^2\theta$ *factor.* $(f/n = 1000, \theta = 30^\circ)$

map [Kozlov 2004] which is essentially FP+P with the PSM parameter used for the perspective warping. The LogP parameterization also provides lower error for point lights.

Table 3 shows the variation in perspective aliasing error measured by the storage factor over all light directions for various algorithms. Standard shadow maps have the highest error, but over all light directions the variation in the error is fairly small. The single shadow map warping algorithms P, Po, and LogP provide lower error for overhead views, but must degenerate to standard shadow maps when the light moves behind or in front of the viewer. This leads to a huge variation in error that makes these algorithms more difficult to use. The table shows that in contrast to Po, our improved shaping function for the warping parameter keeps the maximum error of P below that of a standard shadow map. Even though LogP has a much lower minimum error than P, it ramps off to a uniform parameterization slightly faster than P and the extremely high error of the uniform parameterization dominates the mean. However, it can be seen from Figure 15 that LogP provides significant improvement over P for almost the entire range of $\gamma \in [\theta, 90^\circ]$.

Face partitioning leads to much lower variation in error over all light directions. Coordinate frame adjustment and face splitting reduces shearing error not accounted for by the storage factor, which causes a slight increase in the storage factor but an overall decrease in actual error. The FP*+LogP algorithms have much lower error than the FP*+P algorithms due to the better parameterization.

As with a single shadow map, z-partitioning with a uniform parameterization has the least variation in error over all light directions. Adding warping reduces the error for $\gamma \in [\theta, 90^\circ]$. The minimum error for ZPk+LogP, which occurs for an overhead directional light, is the same for all k. With this light position, increasing the number z-partitions has no effect on the parameterization. This is not the case for uniform and perspective parameterizations. Figure 15 shows that for other light positions increasing k produces drastic reductions in error that then trail off. The benefit of warping is also reduced. For comparison, the error for FP+Log and FPcs+LogP are also shown in Figure 15. A small rise in FPcs+LogP can be seen at $\gamma = \theta$ where a new face partition appears and is split. As the number of z-partitions increases, the error begins to approach that of the FP+LogP. We have shown the error for the ZPk algorithms with $k = 5$ and $k = 7$ because these are the maximum number of shadow maps required for the FP and FPc algorithms and the FPcs algorithm respectively. On average, however,

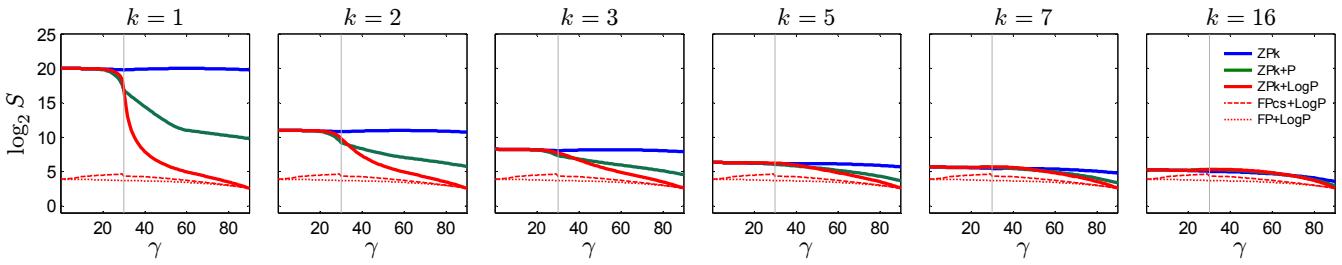

Figure 15: z-partitioning using various parameterizations. *z-partitioning leads to significant error reductions but requires many partitions to converge to the same error as a face partitioning scheme that uses a logarithmic+perspective parameterization. ($f/n = 1024$, $\theta = 30°$)*

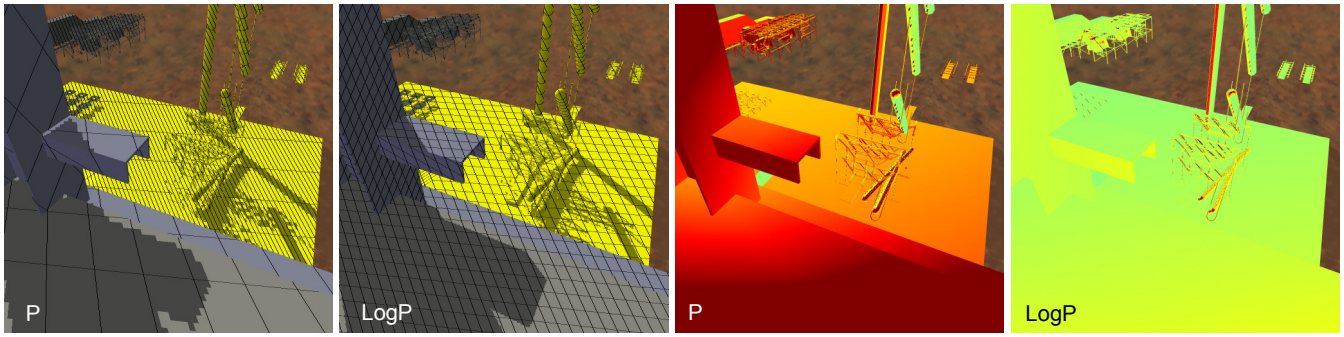

Figure 16: Comparison with single shadow map in a power plant model. *Here we compare the P and LogP parameterizations. The image resolution is 512×512 and the shadow map resolution is 1024×1024. Grid lines are shown for every 10 texels. ($f/n = 500$, $\theta = 30°$)*

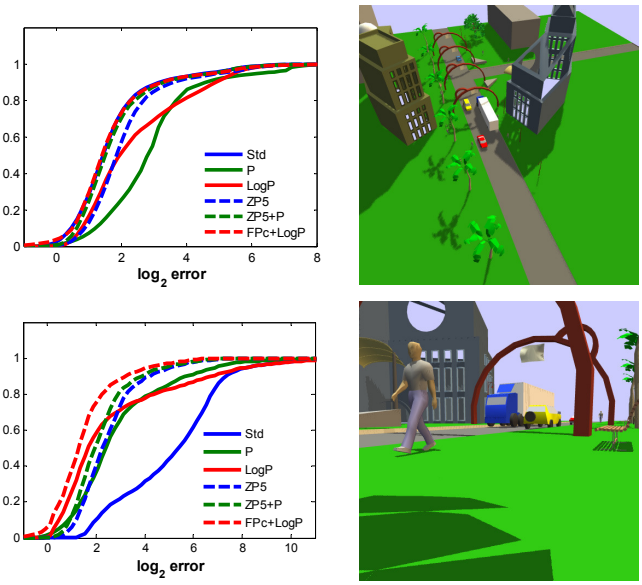

Figure 17: Cumulative aliasing error distribution over randomly sampled light directions for several algorithms. *The graphs on the left were generated for the views on the right. The area of the projected texel was used for the error metric. The actual benefit of warping and partitioning methods depends on the location and orientation of surfaces within the view frustum.*

the number of shadow maps used by FP, and FPc is 3.6, and for FPcs the average number is only 3.8.

The statistics reported in Table 3 show scene-independent maximum perspective aliasing error over all light directions. To get an idea of how well this correlates with the actual aliasing error in the image for a real scene we uniformly sampled the hemisphere of light directions above the scene and created a histogram of the error in the image. Figure 17 shows the cumulative distribution of the error. The value y at a point x on the curve is the fraction of pixels with error less than x. The distribution of error depends on the positions of the surfaces within the view frustum. We see that the single shadow map algorithms tend to have higher error spread out over a wider range. When the surfaces are far from the eye, as in the first view shown in Figure 17, warping and partitioning of any kind may produce little benefit over standard shadow maps. In fact, the warping algorithm P produces error that is significantly worse than a standard shadow map for this view. When the surfaces are closer to the eye, as in the second view, FPc+LogP can produce significantly less error than other parameterizations.

From the analysis in this section we can see that an FP scheme with a LogP parameterization produces the lowest error. Compared to a ZP scheme using a comparable number of shadow maps and a uniform or perspective parameterization, FP*+LogP algorithms provide more modest error reductions. The LogP parameterization is most useful for a single shadow map in applications where the light direction does not approach the view direction (e.g. shadows from the sun around mid-day in a driving simulator), with a ZP scheme using a small number of partitions, and for omnidirectional point lights. For omnidirectional lights, z-partitioning must be applied to all the faces of a standard cube map, or to the side faces in the case of a warped cube map. Using an FP scheme with a LogP parameterization requires much fewer shadow maps.

7.1 Limitations

LogPSMs inherit some of the limitations of sample redistribution techniques. Warping creates a non-uniform distribution of depth values, which can make it more difficult to select a suitable constant bias. This is less of an issue

for slope scaled bias. Warping also tends to increase the instability of texture coordinate derivative calculations on surfaces nearly parallel to the light (some noise can be seen in Figure 12). Handling the shearing artifacts created by face partitioning requires up to 2 extra shadow maps and added complexity. Most importantly, logarithmic rasterization is not available on current GPUs. It can be simulated with a fragment program, but it is considerably slower than linear rasterization.

Conclusion

We have presented the logarithmic perspective parameterization and have showed how it can be used to extend existing algorithms. With proper hardware support, LogPSMs would have the same good performance as the algorithms upon which they are based but they provide lower error. For future work we would like to investigate the use of the logarithmic perspective parameterizations with algorithms such as adaptive shadow maps, which can handle projective aliasing.

References

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

Arvo, J. 2004. Tiled shadow maps. In *Proceedings of Computer Graphics International 2004*, IEEE Computer Society, 240–247.

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.

Chong, H., and Gortler, S. 2006. Scene optimized shadow mapping. Tech. Rep. TR-11-06, Harvard University.

Chong, H. 2003. *Real-Time Perspective Optimal Shadow Maps.* Senior Thesis, Harvard University.

Crow, F. C. 1977. Shadow algorithms for computer graphics. *ACM Computer Graphics 11*, 3, 242–248.

Donnelly, W., and Lauritzen, A. 2006. Variance shadow maps. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 161–165.

Engel, W. 2007. Cascaded shadow maps. In *ShaderX*^{5}, W. Engel, Ed. Charles River Media, 197–206.

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

Forsyth, T. 2006. Making shadow buffers robust using multiple dynamic frustums. In *ShaderX*^{4}, W. Engel, Ed. Charles River Media, 331–346.

Giegl, M., and Wimmer, M. 2007. Queried virtual shadow maps. In *Proceedings of ACM SIGGRAPH 2007 Symposium on Interactive 3D Graphics and Games*, ACM Press, 65–72.

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.*

Johnson, G. S., Lee, J., Burns, C. A., and Mark, W. R. 2005. The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Trans. Graph. 24*, 4, 1462–1482.

Kozlov, S. 2004. Perspective shadow maps: Care and feeding. In *GPU Gems*, R. Fernando, Ed. Addison-Wesley, 214–244.

Lefohn, A. 2006. Glift: Generic data structures for graphics hardware. Ph.D. Dissertation, Dept. Comput. Sci., Univ. of California Davis.

Lloyd, B., Tuft, D., Yoon, S., and Manocha, D. 2006. Warping and partitioning for low error shadow maps. In *Proceedings of the Eurographics Symposium on Rendering 2006*, Eurographics Association, 215–226.

Lloyd, D. B., Govindaraju, N. K., Molnar, S., and Manocha, D. 2007. Practical logarithmic rasterization for low-error shadow maps. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, Eurographics Association.

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., Wales, C., and Moule, K. 2001. Incremental and hierarchical hilbert order edge equation using polygon rasteization. *Eurographics Workshop on Graphics Hardware*, 65–72.

Reeves, W., Salesin, D., and Cook, R. 1987. Rendering antialiased shadows with depth maps. In *Computer Graphics (ACM SIGGRAPH '87 Proceedings)*, vol. 21, 283–291.

Sen, P., Cammarano, M., and Hanrahan, P. 2003. Shadow silhouette maps. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (July), 521–526.

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.

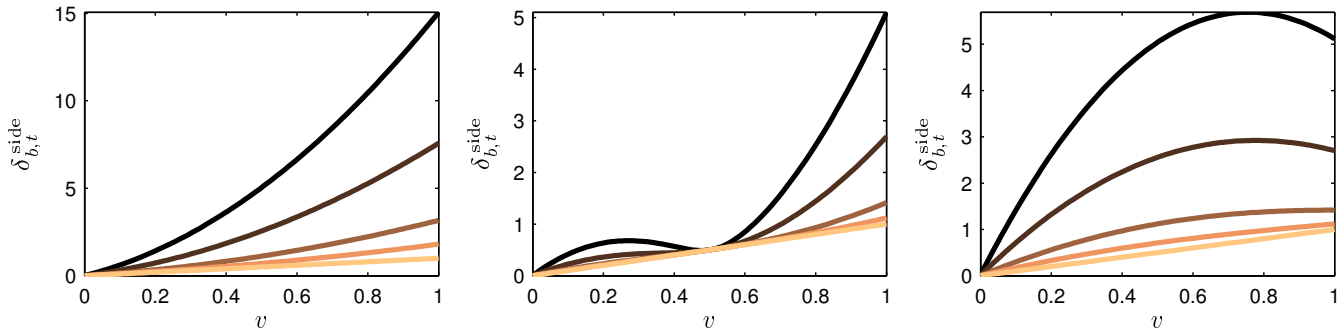

Figure 18: Error bound spacing distribution for t on a side face. *These graphs show $\delta^{side}_{b,t}$ for the light at $l_y = \kappa(y_1 - y_0)$ and $l_z = \sigma(y_1 - y_0)$. From left to right, $\kappa = -0.5, 0.5, 1$. From dark to light, $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The frustum parameters are $\theta = 45°$, $n_e = 1$, and $f_e = 1000$.*

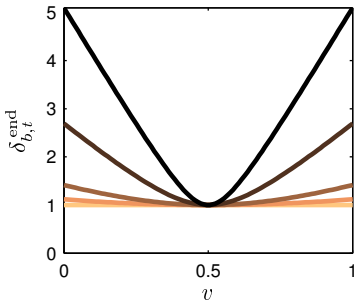

Figure 19: Error bound spacing distribution on an end face. *This graph shows $\delta^{end}_{b,t}$ for the light at $l_y = 0$ and $l_z = \sigma(y_1 - y_0)$. From dark to light, $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The $\cos\theta$ term has been factored out.*

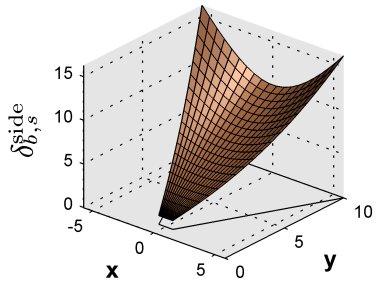

Figure 20: Error bound spacing distribution for s on a side face. *This graph shows $\delta^{side}_{b,s}$ with the light centered over the face.*

Zhang, F., Sun, H., Xu, L., and Lun, L. K. 2006. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of ACM International Conference on Virtual Reality Continuum and Its Applications 2006*, ACM SIGGRAPH, 311–318.

Zhang, F., Xu, L., Tao, C., and Sun, H. 2006. Generalized linear perspective shadow map reparameterization. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, ACM Press, New York, NY, USA, 339–342.

Appendix A

In this appendix we analyze various shadow map parameterizations for the faces of the view frustum. We begin with the parameterizations F_b for the three varieties of δ_b. We also discuss the uniform, perspective, logarithmic, and logarithmic+perspective parameterizations described in Section 5.

Parameterizations based on δ_b

We begin by looking at some graphs of the various spacing functions δ_b based on the bound for perspective aliasing. Figure 19 shows how $\delta^{\text{end}}_{b,t}$ on the far face changes with light position. The $\cos\phi_{ly}$ term is responsible for the variation in the shape of the function. Close to the face, $\delta^{\text{end}}_{b,t}$ resembles an absolute value function with a rounded tip. As $l_z \to \infty$, $\cos\phi_{ly}$ converges to a constant and the spacing function converges to uniform. Moving the light in y simply translates the function along the v-axis.

Figure 20 shows the shape of $\delta^{\text{side}}_{b,s}$ for a light centered over the face. Note that $\delta^{\text{side}}_{b,s}$ is essentially $\delta^{\text{end}}_{b,t}$ extruded along the y direction and scaled by y.

Figure 18 shows $\delta^{\text{side}}_{b,t}$ for the light at varying heights above the face. When the light is close to the face, the spacing function is an undulating curve with dense sampling near the light and the viewer. Translating the light in y produces a shifted and scaled version of the function with l_y centered over the face. As the light moves away toward infinity, the spacing function converges to a linear function. Once again, it is the $\cos\phi_{ly}$ term close to the face that accounts for the variation in the shape of the spacing function.

From Equation 47 we can see that $R_b = \rho_b$ because $\mathrm{d}F_b/\mathrm{d}v = \rho_b\delta_b$. We note that $\rho^{\text{side}}_{b,s}$ varies with v. Therefore we take $R^{\text{side}}_{b,s} = \max_v(\rho^{\text{side}}_{b,s}(v))$. Figure 21 shows R_b for the three varieties of δ_b. In general, R_b is smaller when the light is close to the plane containing the face. It also falls off as the light moves to either side of the face. This is due to the effect of the $\cos\phi_l$ term. R_b is related to the integral of $\cos\phi_l$ over the face. When the light is close to face or off to the side, the $\cos\phi_l$ term becomes smaller. It is largest for points directly under the light. For an overhead directional light $\cos\phi_l = 1$ everywhere. It is for this position that R_b reaches its maximum value. Therefore, the maximum critical resolution factors R_b can be computed as $\lim_{l_z\to\infty}\rho_b$, or equivalently, by plugging $\cos\phi_l = 1$ into Equations 58, 61,

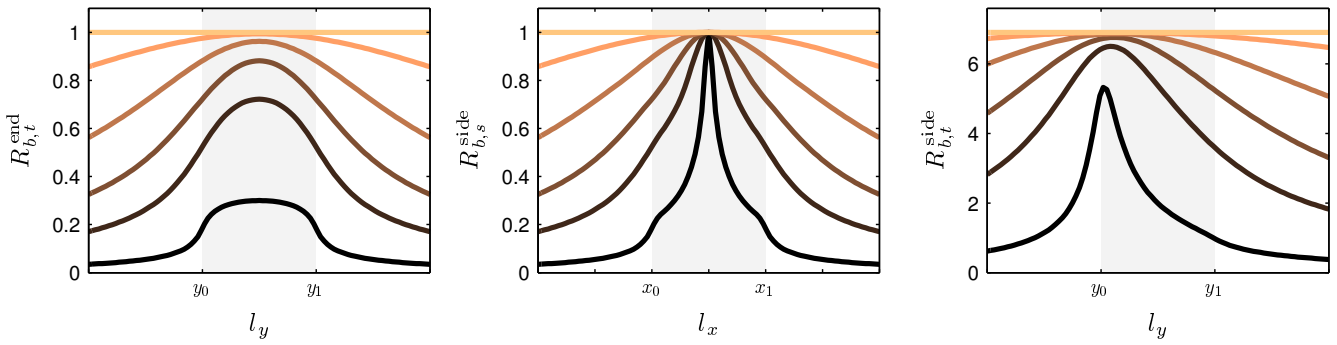

Figure 21: Critical resolution factor for error bound parameterizations. *From left to right, $R^{end}_{b,t}$, $R^{side}_{b,s}$, and $R^{side}_{b,t}$ for the light at varying positions. The plots show $l_z = \sigma(y_1 - y_0)$, where from dark to light, $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The grayed out region indicates light positions over the face. The frustum parameters are $\theta = 45°$, $n_e = 1$, and $f_e = 1000$. The $\cos\theta$ term has been factored out of $R^{end}_{b,t}$ and $R^{side}_{b,s}$.*

and 62, and computing the corresponding ρ_b:

$$\mathsf{R}^{\text{end}}_{b,t} = \frac{1}{\cos\theta} \tag{87}$$

$$\mathsf{R}^{\text{side}}_{b,s} = \frac{1}{\cos\theta} \tag{88}$$

$$\begin{aligned}\mathsf{R}^{\text{side}}_{b,t} &= \frac{y_0 \log(y_1/y_0)}{W_e \cos\theta} = \frac{\log(f_e/n_e)}{2\tan\theta\cos^2\theta} \\ &= \frac{\log(f_e/n_e)}{\sin 2\theta}\end{aligned} \tag{89}$$

We have substituted $W_e = 2n_e \tan\theta$. Thus the critical resolution factors are at most $O(1)$ for an end face and $O(\log(f_e/n_e))$ for a side face.

Uniform parameterization

A standard shadow map uses a parameterization that has a uniform spacing distribution function:

$$\delta_{un} = 1 \tag{90}$$

$$\rho_{un} = 1. \tag{91}$$

From Equation 47 we get for an end face:

$$R^{\text{end}}_{un,t} = \max_v \left(\frac{1}{\delta^{\text{end}}_{b,t}(v)} \right) = \max_v \left(\frac{\cos\phi_{ly}(v)}{\cos\theta} \right) \tag{92}$$

Figure 22 shows R_{un} on an end face and a side face for various light positions. Note that R_{un} reaches its maximum whenever the light position is over the face. That is because the $\cos\phi_{ly}$ factor reaches its maximum for points where $y(v) = l_y$. Comparing Figure 22 to Figure 21, we see that when the light is not directly over the face, the shapes of the graphs are nearly identical, but on the side face, the magnitudes of the plots differ significantly. This difference arises from the $y(v)$ term in $\delta^{\text{side}}_{b,s}$ and $\delta^{\text{side}}_{b,t}$, which varies by a factor of f_e/n_e over the face.

The various R_{un} can be computed most simply by setting the $\cos\phi_l$ term in δ_b to 1 and applying the equivalent of Equation 92 for the appropriate face. On side faces, the remaining $y(v)$ terms of δ^{side}_b have their minimum at $y(v) = y0$.

$$\mathsf{R}^{\text{end}}_{un,t} = \frac{1}{\cos\theta} \tag{93}$$

$$\mathsf{R}^{\text{side}}_{un,s} = \frac{y_1}{y_0\cos\theta} = \frac{f_e}{n_e\cos\theta} \tag{94}$$

$$\begin{aligned}\mathsf{R}^{\text{side}}_{un,t} &= \frac{(y_1 - y_0)}{W_e\cos\theta} = \frac{f_e - n_e}{2n_e\tan\theta\cos^2\theta} \\ &= \frac{(f_e/n_e) - 1}{\sin 2\theta}.\end{aligned} \tag{95}$$

The maximum critical resolution for the uniform parameterization is identical to that of the error bound parameterization for the end faces.

Perspective parameterization

Existing warping methods use perspective projections. The normalized spacing distribution functions on a side face for the perspective parameterization $\mathbf{F}_p$ in Equation 75 can be computed by taking the multiplicative inverse of the derivatives of $\mathbf{F}_{p,s}$ and $\mathbf{F}_{p,t}$ w.r.t. u and v, respectively:

$$\begin{aligned}\delta^{\text{side}}_{p,s} &= \frac{y(v) + a}{p_0\,(x_1(y_1) - x_0(y_1))} \\ &= \frac{(y(v) + a)}{(y_1 + a)}\end{aligned} \tag{96}$$

$$\begin{aligned}\delta^{\text{side}}_{p,t} &= -\frac{(y(v) + a)^2}{p_2(y_1 - y_0)} \\ &= \frac{(y(v) + a)^2}{(y_0 + a)(y_1 + a)}.\end{aligned} \tag{97}$$

$\delta^{\text{side}}_{p,s}$ is constant in u and increases linearly in v, while $\delta^{\text{side}}_{p,t}$ is quadratic in v.

Figure 23 shows the optimal $R^{\text{side}}_{p,s}$, $R^{\text{side}}_{p,t}$, and S^{side}_p. The optimal $R^{\text{side}}_{p,s}$ is computed as:

$$\begin{aligned}\min_a R^{\text{side}}_{p,s} &= \min_a \max_{(u,v)\in\mathcal{F}} \left(\frac{\delta^{\text{side}}_{p,s}}{\delta^{\text{side}}_{b,s}} \right) \\ &= \min_a \max_{(u,v)\in\mathcal{F}} \left(\frac{y1}{(y1 + a)} \frac{(y(v) + a)}{y(v)} \frac{\cos\phi_{lx}(u)}{\cos\theta} \right).\end{aligned} \tag{98}$$

The maximum in $R^{\text{side}}_{p,s}$ is achieved when $y(v) = y_0$. $R^{\text{side}}_{p,s}$ is minimized when $a = 0$, which leaves only the $\cos\phi_l x/\cos\theta$

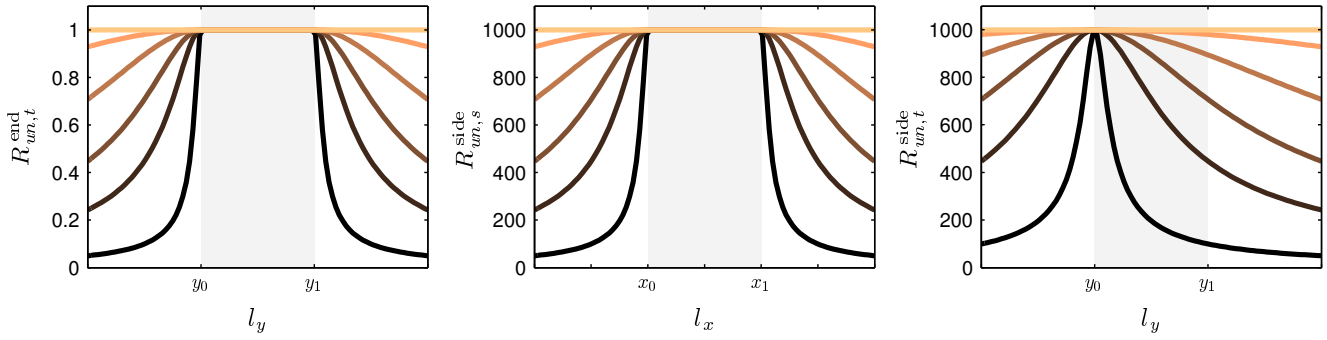

Figure 22: Critical resolution factor for uniform parameterization. *From left to right, $R^{end}_{un,t}$, $R^{side}_{un,s}$, and $R^{side}_{un,t}$ for various light positions. For all graphs $l_z = \sigma(y_1 - y_0)$, where from dark to light, the plots are for $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The grayed out region indicates light positions over the face. The frustum parameters are $\theta = 45°$, $n_e = 1$, and $f_e = 1000$. The $\cos\theta$ term has been factored out of $R^{end}_{un,t}$ and $R^{side}_{un,s}$.*

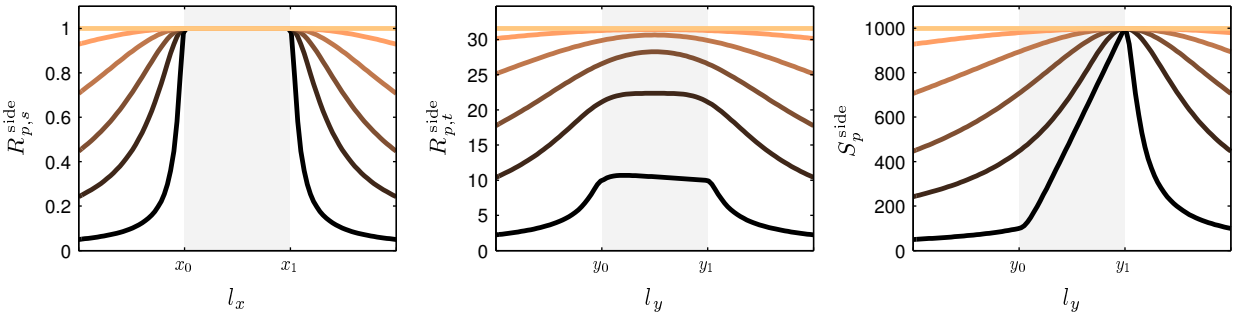

Figure 23: Optimal critical resolution factors for perspective parameterization. *From left to right $R^{side}_{p,s}$, $R^{side}_{p,t}$, and S^{side}_{p} for various light positions. When parameterizing with a perspective matrix, $\delta^{side}_{p,s}$ and $\delta^{side}_{p,t}$ are coupled. S^{side}_{p} combines the errors in both directions so that the optimal perspective parameter can be found considering both directions together. For all graphs $l_z = \sigma(y_1 - y_0)$, where from dark to light, the plots are for $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The grayed out region indicates light positions over the face. The frustum parameters are $\theta = 45°$, $n_e = 1$, and $f_e = 1000$. The $\cos\theta$ term has been factored out of $R^{side}_{p,s}$.*

term in Equation 98. Geometrically, setting $a = 0$ causes the face to be stretched out to fill the entire shadow map, maximizing the use of the available resolution. With $a = 0$, Equation 98 is essentially the same as Equation 92. The optimal a parameter for $R^{\text{side}}_{p,t}$ varies with the position of the light. For a directional light directly above the face, $a_{\text{opt}} = \sqrt{y_0 y_1}$, the optimal parameter for LiSPSMs. The maximum occurs at either $y(v) = y_0$ or $y(v) = y_1$. Using these optimal parameters for s and t for the overhead light case gives the optimal $\mathsf{R}^{\text{side}}_p$:

$$\min_a \mathsf{R}^{\text{side}}_{p,s} = \frac{1}{\cos\theta} \tag{99}$$

$$\begin{aligned}\min_a \mathsf{R}^{\text{side}}_{p,t} &= \frac{y_0(y_1 - y_0)}{W_e\sqrt{y_0 y_1}\cos\theta} = \frac{(f_e - n_e)}{2\tan\theta\cos^3\theta\sqrt{n_e f_e}} \\ &= \frac{f_e/n_e - 1}{\sin 2\theta\sqrt{f_e/n_e}\cos\theta} \\ &\approx O(\sqrt{f_e/n_e}).\end{aligned} \tag{100}$$

The parameterization $\mathbf{F}_p$ is usually implemented with a projective matrix, in which case the parameter a is the same for s and t. This means that $R^{\text{side}}_{p,s}$ and $R^{\text{side}}_{p,t}$ can not be optimized separately as we have done. Rather they should be optimized simultaneously by optimizing the storage factor S^{side}_p. Lloyd et al. [2006] noted that for overhead directional lights there exists a range of parameter values for which S^{side}_p is minimal. Figure 24 shows that a range of optimal parameter values for S^{side}_p also exists for point lights. S, however, does not measure the error due to shearing. Shearing decreases with increasing a. Therefore it is advisable to choose the largest a possible on the equivalent range. For an overhead directional light this corresponds to the LiSPSM parameter. Using this value we can compute $\mathsf{S}^{\text{side}}_p$:

$$\begin{aligned}\min_a \mathsf{S}^{\text{side}}_p &= \frac{y_1 - y_0}{W_e\cos^2\theta} = \frac{(f_e - n_e)}{2n_e\tan\theta\cos^2\theta} \\ &= \frac{(f_e/n_e - 1)}{\sin 2\theta} \\ &\approx O(f_e/n_e).\end{aligned} \tag{101}$$

The optimal range for a includes $a = 0$. If $a = 0$, then $R^{\text{side}}_{p,s}$ is minimized and most of the error is in t.

Logarithmic parameterization

We have now shown that the uniform parameterization gives error for end faces on the same order as that of the error bound parameterization. The perspective parameterization can do the same for the s on side faces. We will now show that a logarithmic parameterization can do the same for t. The normalized spacing distribution for Equation 77 is:

$$\delta^{\text{side}}_{\log,t} = \log\left(\frac{y_1 + a}{y_0 + a}\right)\frac{y + a}{y_1 - y_0} \tag{102}$$

This spacing distribution is linear, as is that of the $\delta^{\text{side}}_{b,t}$ for a directional light (see Figure 18). Figure 25 shows $R^{\text{side}}_{\log,t}$.

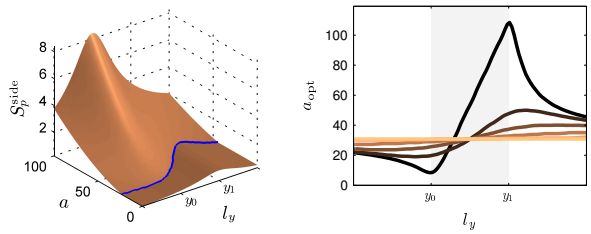

Figure 24: Optimal perspective parameter. (Left) *When the errors in the s and t are combined, a range of parameters (those below the curve) produce the same storage factor S_p^{side} for each light position ($l_y = 0.5y_1$.)* (Right) *The maximum parameter value in optimal range for $l_z = \sigma(y_1 - y_0)$, where from dark to light $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The frustum parameters are $\theta = 45°$, $n_e = 1$, and $f_e = 1000$.*

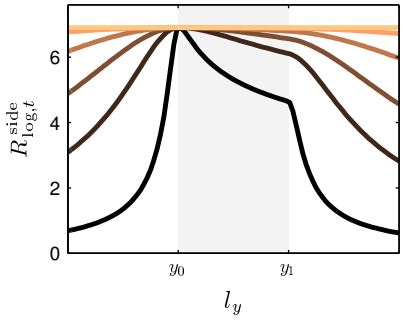

Figure 25: Critical resolution factor for logarithmic parameterization. *For all graphs $l_z = \sigma(y_1 - y_0)$, where from dark to light, the plots are for $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The grayed out region indicates light positions over the face. The frustum parameters are $\theta = 45°$, $n_e = 1$, and $f_e = 1000$.*

Because $\delta_{\log,t}^{\text{side}}$ is not the best match for the $\cos\phi_l$ factor, the error is not as low as it could be when the light is close to the face, but it is on the same order as $R_{b,t}^{\text{side}}$. When the $\cos\phi_l$ factor in $\delta_{b,t}^{\text{side}}$ becomes 1 for a directional light, the optimal $a_{\text{opt}} = 0$ gives a perfect match for the $y(v)$ term. With this parameter we get:

$$\min_a \mathsf{R}_{\log,t}^{\text{side}} = \frac{y_0 \log(y_1/y_0)}{W_e \cos\theta} = \frac{\log(f_e/n_e)}{\sin\ 2\theta}, \tag{103}$$

which is the same as $\mathsf{R}_{b,t}^{\text{side}}$.

Logarithmic + perspective parameterization

The general spacing distribution for $F_{lp,t}$ is found by computing $(\mathrm{d}F_{lp,t}/\mathrm{d}v)^{-1}$:

$$\delta_{lp,t} = \frac{(y(v)+a)\Big((y(v)+a)(c_1p_2+c_2)+c_1p_3\Big)}{c_0c_1p_3(y_1-y_0)}. \tag{104}$$

$\delta_{lp,t}$ has two real roots (in terms of $y(v)$):

$$\lambda_0 = -a, \qquad \lambda_1 = -\left(a + \frac{c_1p_3}{c_1p_2+c_2}\right). \tag{105}$$

If we choose the constants c_1 and c_2 such that $\lambda_1 = -b$, we can write $\delta_{lp,t}$ as:

$$\delta_{lp,t} \sim (y(v)+a)(y(v)+b). \tag{106}$$

To compute c_1 and c_2 we need an additional constraint, which comes from the requirement that $F_{lp,t}(y_0) = 0$. This constraint is satisfied when the argument of the logarithm, $c_1F_{p,t}(v) + c_2$, is 1. Solving for c_1 and c_2 we get:

$$c_1 = -\frac{(y_0+a)(a-b)}{(y_0+b)p_3} \tag{107}$$

$$c_2 = \frac{(y_0+a)(ap_2-bp_2+p_3)}{(y_0+b)p_3}. \tag{108}$$

Plugging these values into Equation 104 gives us the normalized spacing distribution:

$$\delta_{lp,t} = -\frac{(y(v)+a)(y(v)+b)}{c_0(a-b)(y_1-y_0)}. \tag{109}$$

The constant c_0 can be computed from the constraint that $F_{lp,t}(y_1) = 1$:

$$c_0 = \frac{1}{\log\left(\frac{(y_0+a)(y_1+b)}{(y_0+b)(y_1+a)}\right)}. \tag{110}$$

When $b \to \infty$ the constants converge to those given in Section 5.3 and $\delta_{lp,t}$ converges to a linear spacing distribution. Figure 26 shows the optimal $S_{lp,t}^{\text{side}}$ with $b = \infty$. The a parameter also effects $R_{lp,s}^{\text{side}}$ and thus must be optimized simultaneously with $R_{lp,t}^{\text{side}}$ by optimizing $S_{lp,t}^{\text{side}}$. We find the optimal a parameter numerically. It appears that the optimal parameter is always $a = 0$. Note that when the light is over the face, $R_{lp,t}^{\text{side}}$ the error is higher than $R_{\log,t}^{\text{side}}$. However $\mathsf{R}_{\log,t}^{\text{side}}$ is still the same.

When b is finite, we get a parabola. Using two parabolas we can get a better fit to the curved portions of $\delta_{b,t}^{\text{side}}$ (see Figure 26). Fitting two parabolas, however, is more expensive because it requires that the face be split into two partitions with each partition parameterized separately. The optimal split point y_s is difficult to compute, but $y_s = l_y$ produces good results that are close to optimal. Figure 26 also shows the optimal $S_{lp,t}^{\text{side}}$ with $y_s = l_y$, $a = 0$, and b free. The curves for $R_{lp,t}^{\text{side}}$ are very similar to those for $R_{b,t}^{\text{side}}$. One disadvantage of this parameterization is that a closed-form solution for the optimal parameter b_{opt} for each parabola does not exist. Therefore b_{opt} must be computed numerically, which is somewhat involved. We would like to further explore the additional degrees of freedom in $\mathbf{F}_{lp}$ as future work.

Appendix B

In this section we analyze the error for parameterizations covering the entire view frustum. The error analysis for point lights that we performed on the frustum faces can be extended for analyzing single shadow map methods. We restrict our attention here, however, to directional lights in order to simplify the discussion. These methods are typically used for directional lights anyway, or for spot lights with a fairly narrow field of view.

We first derive equations that can be used to analyze the error for varying light directions. These equations are similar to those used by Zhang et al. [2006b]. The main difference is that they analyze error along the view direction while we evaluate the error along the faces.

Figure 27 shows the light image plane and light frustum surrounding the view frustum. We parameterize the light

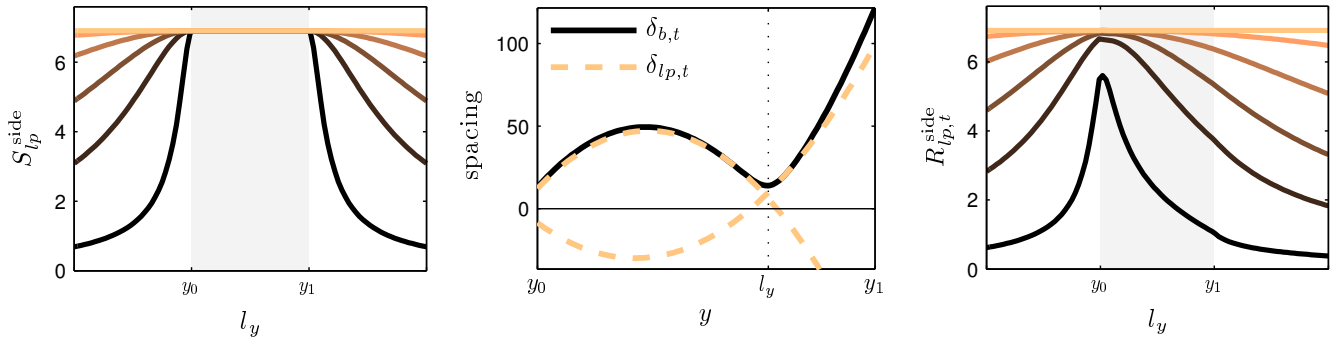

Figure 26: Critical resolution and storage factors for logarithmic+perspective parameterizations. (Left) *Storage factor for the linear form of $\delta_{lp,t}^{side}$ that we use for LogPSMs.* (Middle) *The logarithmic + perspective parameterization can also generate parabolic spacing functions that can give a good fit to $\delta_{b,t}^{side}$.* (Right) *The two parabola form gives a storage factor that approaches that of δ_b^{side}. For the plots on the left and the right, $l_x = 0$ and $z = \sigma y_1$, where from dark to light $\sigma = 0.1, 0.2, 0.5, 1, \infty$. The grayed out region indicates light positions over the face. The frustum parameters are $\theta = 45°$, $n_e = 1$, and $f_e = 1000$.*

direction by the angle γ between the light and view directions. We will consider only $\gamma \in [0°, 90°]$. For this range, the maximum perspective error occurs on the back faces of the view frustum. The near face is back-facing for all $\gamma < 90°$. Both side faces are back-facing when $\gamma < \theta$, while only one of them is for $\gamma > \theta$. We compute the error distribution for points on the light image plane corresponding to these faces as $(\mathrm{d}F/\mathrm{d}v)^{-1}/\delta_b$. The various terms of δ_b can be computed using simple geometry on Figure 27:

$$W_{ly} = W'_n + W'_{s2} + \begin{cases} W'_{s1} & \gamma \in [0, \theta] \\ 0 & \gamma \in [\theta, 90°] \end{cases} \tag{111}$$

$$W_{lx} = (n' + W_{ly}) \begin{cases} \frac{W_f}{n'} & \gamma \in [0, \theta] \\ \frac{W_f}{n' + W'_{s1}} & \gamma \in [\theta, 90°] \end{cases} \tag{112}$$

$$d_e(v) = n_e + (f_e - n_e) \max\left(\frac{y(v_{n0}) - y(v)}{W'_{s1}}, 0, \frac{W_{ly} - y(v)}{W'_{s2}}\right) \tag{113}$$

$$y(v) = W_{ly} v \tag{114}$$

$$v_{n0} = \begin{cases} \frac{W'_{s1}}{W_{ly}} & \gamma \in [0, \theta] \\ 0 & \gamma \in [\theta, 90°] \end{cases} \qquad v_{n1} = v_{n0} + \frac{W'_n}{W_{ly}} \tag{115}$$

$$W'_n = W_n \cos\gamma, \tag{116}$$

$$W'_{s1} = W_s\,(1 - \cos(\theta - \gamma)) \tag{117}$$

$$W'_{s2} = W_s \sin(\theta - \gamma) \tag{118}$$

$$W_n = W_e, \qquad W_f = W_n \frac{f_e}{n_e}, \qquad W_s = \frac{f_e - n_e}{\cos\theta}. \tag{119}$$

Because we are using a directional light source and the light image plane is perpendicular to the light direction, both the n_l/d_l and $\cos\phi_l$ terms are 1. If we define $y_0 = 0$, $y_1 = W_{ly}$, and $a = n'$ we can use the spacing distribution functions computed for the various parameterizations in Appendix A for frustum parameterizations.

Armed with these equations we can now analyze the error distribution of the uniform, perspective, and logarithmic perspective parameterizations for a single frustum. We will assume throughout this discussion that $r_s = r_i$ and $r_t = r_j$. We will also drop the $\cos\theta$ term from δ_b so that the dependence of the error on a function of f_e/n_e can be more easily observed. Because the warping parameter n' which controls $\mathbf{F}_p$ and $\mathbf{F}_{lp}$ can diverge to ∞, we parameterize n' with η,

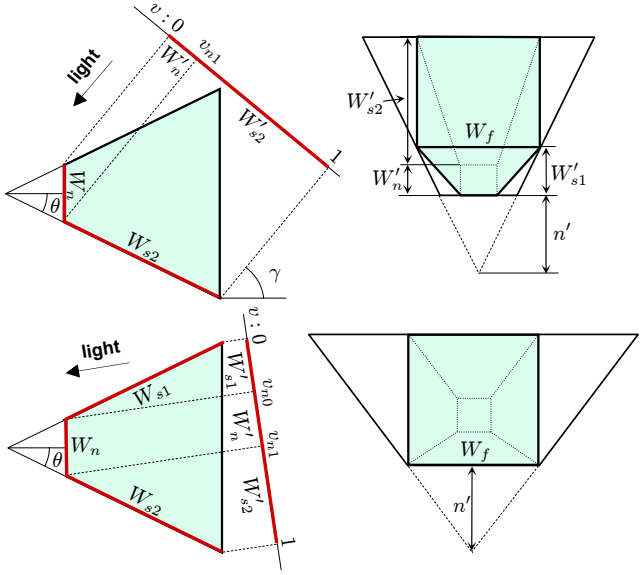

Figure 27: Parameterizing the entire frustum. *(Left) A side view of the view frustum. The back faces of the view frustum are projected onto the light image plane, which is oriented perpendicular to the light direction. (Right) The light's view of the view frustum. The perspective warping frustum surrounding the view frustum is shown in black. On the top row $\gamma > \theta$ and on the bottom row $\gamma < \theta$.*

which is defined over the finite range $[-1, 1]$. As given by Lloyd et al. [2006], η is parameterization of n' that corresponds to $n' = \infty$ at $\eta = -1$, $n' = n_e + \sqrt{n_e f_e}$ at $\eta = 0$, and $n' = n_e$ at $\eta = 1$. We modify their equations slightly for use with varying γ:

$$n'(\eta) = \frac{W_{ly}}{\alpha - 1} \begin{cases} \frac{\sqrt{\alpha}+1-\eta(\alpha-1)}{\eta+1} & \eta < 0 \\ \frac{\sqrt{\alpha}+1}{\eta\sqrt{\alpha}+1} & \eta \geq 0 \end{cases} \tag{120}$$

$$\alpha = \frac{f_e}{n_e}. \tag{121}$$

Figure 28 shows the error distribution using $\mathbf{F}_p$ and $\mathbf{F}_{lp}$ over all η. The graphs are shown over all v. $F_{p,s}(u, v)$ is constant in u, which is not shown. Let us consider first the case of an overhead light with $\gamma = 90°$. The value of $\widetilde{M}_{p,s}$ (and $\widetilde{M}_{lp,s}$ because $F_{lp,s} = F_{p,s}$) attains its maximum at $v = 0$, decreases rapidly, and then descends more slowly to its minimum value at $v = 1$. At $\eta = 1$ the warping frustum matches

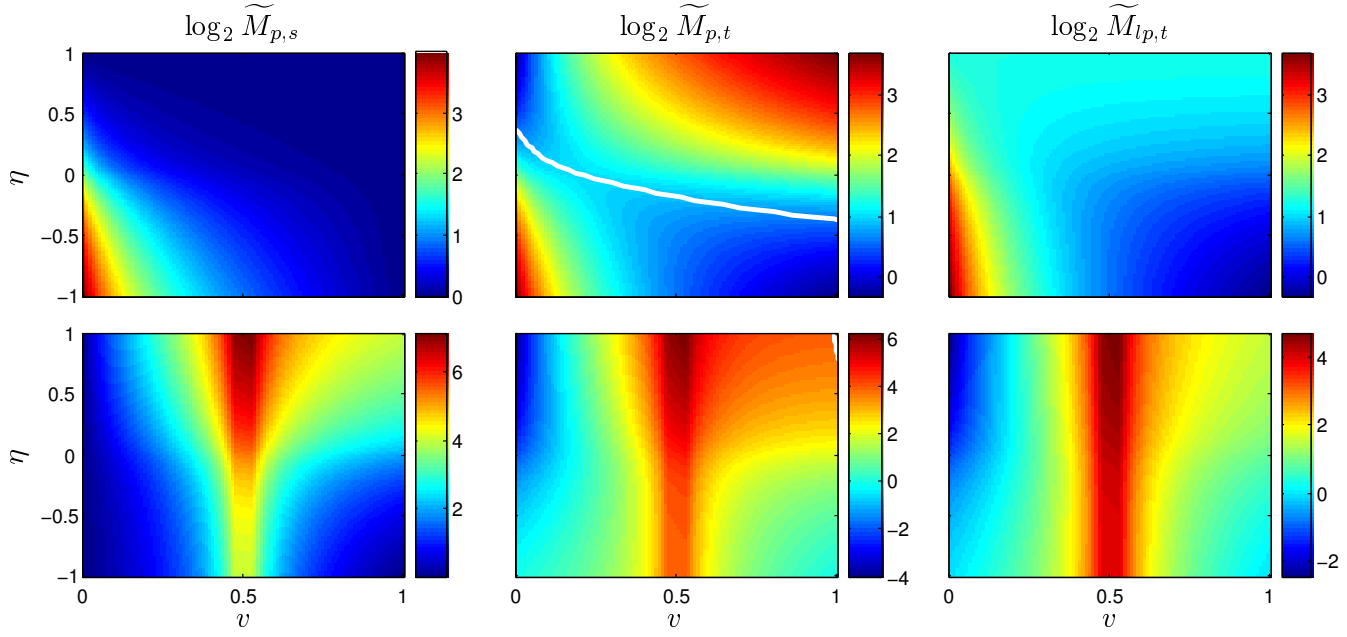

Figure 28: Error distribution for varying warping parameter. *(Top) $\gamma = 90°$. (Bottom) $\gamma = 0°$. The white line on the graph for $\widetilde{M}_{p,t}$ is the location of the minimum error over v as a function of η. The frustum parameters are $n = 1$, $f = 16$, and $\theta = 30°$.*

that of the view frustum and the error is constant and minimal over all v. $\widetilde{M}_{p,s}$ shows a wider range of behavior. At $\eta = -1$, $\widetilde{M}_{p,t}$ degenerates to a uniform parameterization and looks very much like $\widetilde{M}_{p,s}$ with an initial, rapid descent followed by a more gradual one. At $\eta = 1$, $\widetilde{M}_{p,t}$ increases (linearly, in fact) from $v = 0$ to $v = 1$. At $\eta = 0$ the values at $v = 0$ and $v = 1$ are the same and the maximum error over all v is minimal. In general, over some range around $\eta = 0$ the minimum error over v does not occur at the endpoints. For the parameters used to generate Figure 28, this range is approximately $\eta \in [-0.4, 0.4]$. Note that η was derived such that the maximum error over v would decrease linearly along $v = 0$ for $\eta \in [-1, 0]$ and rise again linearly along $v = 1$ for $\eta \in [0, 1]$. This behavior can be seen in the graph. As with $\widetilde{M}_{p,s}$, the maximum of $\widetilde{M}_{p,t}$ is always at $v = 0$, the minimum is at $v = 1$, and at $\eta = 1$ the error is uniform over all v.

Looking at the graphs for the light directly in front of the view frustum ($\gamma = 0°$) we see that the maximum error occurs on the farthest end of near plane at $v = v_{n1}$. This is true for all $\gamma \in [0°, 90°]$. When $\gamma = 90°$, $v_{n1} = 0$. The minimal maximum error is attained for all parameterizations at $\eta = -1$, which is why the single shadow map warping algorithms all degenerate to a uniform parameterization at $\gamma = 0°$.

In order to compute a function that gives us a smooth degeneration to a uniform parameterization as $\gamma \to 0°$ while keeping the error low, we examine the behavior of these functions over varying γ in Figure 29. Based on the analysis of Figure 28, we plot the error at $v = v_{n1}$, $v = 1$, and for $\widetilde{M}_{p,t}$, at the location of the minimum $v = v_{\min}$. The maximum value of all the plots is the critical resolution factor R and the distance between the maximum and minimum captures the variation in error over v. For an overhead light, both R and the variation in $\widetilde{M}$ for $F_{p,s}$ and $F_{lp,t}$ are minimal at $\eta = 1$. As γ decreases, the variation in error increases slightly at $\eta = 1$, but $R_{p,s}$ rises significantly, while $R_{lp,t}$ remains practically the same. At $\gamma = \theta$, there is a rapid shift in R for $\eta \in [0, 1]$. $\widetilde{M}_{p,s}$ also sees a rapid change at $v = v_{n1}$ for η in this range, but the shift in $R_{p,t}$ is more gradual because the maximum error is already high. As γ continues to decrease, the maximum error for all the functions grows steadily until it is higher than that of a uniform parameterization for all $\eta \neq 0$. From this we can see that as γ decreases, it is important that η be close to 0 by the time $\gamma = \theta$. As γ continues to decrease, η should eventually decrease to -1.

Figure 30 shows the error over $\gamma \in [0°, 90°]$ for various shaping functions for the warping parameter. When each parameterization uses its optimal parameter without any shaping, the error for $\gamma < \theta$ is very high. The LiSPSM algorithm uses a $1/\sin\gamma$ factor to modulate the optimal parameter. The figure shows that the effect of this shaping function is relatively minor for $\gamma \in [\theta, 90°]$, which is desirable for keeping the error low. But for $\gamma \in [\theta, 90°]$ the falloff is not fast enough to escape the increase in error in $\widetilde{M}_s$ for $\eta \neq -1$. $\widetilde{M}_{lp,s}$ has even higher error because η_{lp} is further from -1 than η_p. If we compute the optimal parameter with respect to S, we see that for $\gamma \in [\theta, 90°]$ that η is the same as with no shaping at all. At $\gamma = \theta$, η suddenly jumps to 0 and then decreases almost linearly to -1 as γ approaches a point somewhere between 0 and θ. The optimal S_p and S_{lp} never rise above S_{un}. One problem with the S-optimal warping parameter is that S can change very rapidly near $\gamma = \theta$.

Computing the parameter than minimizes S_p and S_{lp} is involved. Instead we propose the simple function in Equation 79 that can nearly replicate the S-optimal warping parameter, while giving the user better control over the shape of S curve. This function provides a linear transition from -1 to η_b on the interval $\gamma \in [\gamma_a, \gamma_b]$. From there it provides a smooth transition to η_c over $\gamma \in [\gamma_b, \gamma_c]$. We choose $\gamma_b = \theta$ because this gives us precise control over η at the point where the error functions begin to change rapidly. Based on observations of the optimal S curves over the typical range of θ and f_e/n_e, we choose $\gamma_a = \theta/3$ for $\mathbf{F}_p$ and $\gamma_a = \theta/2$ for $\mathbf{F}_{lp}$.

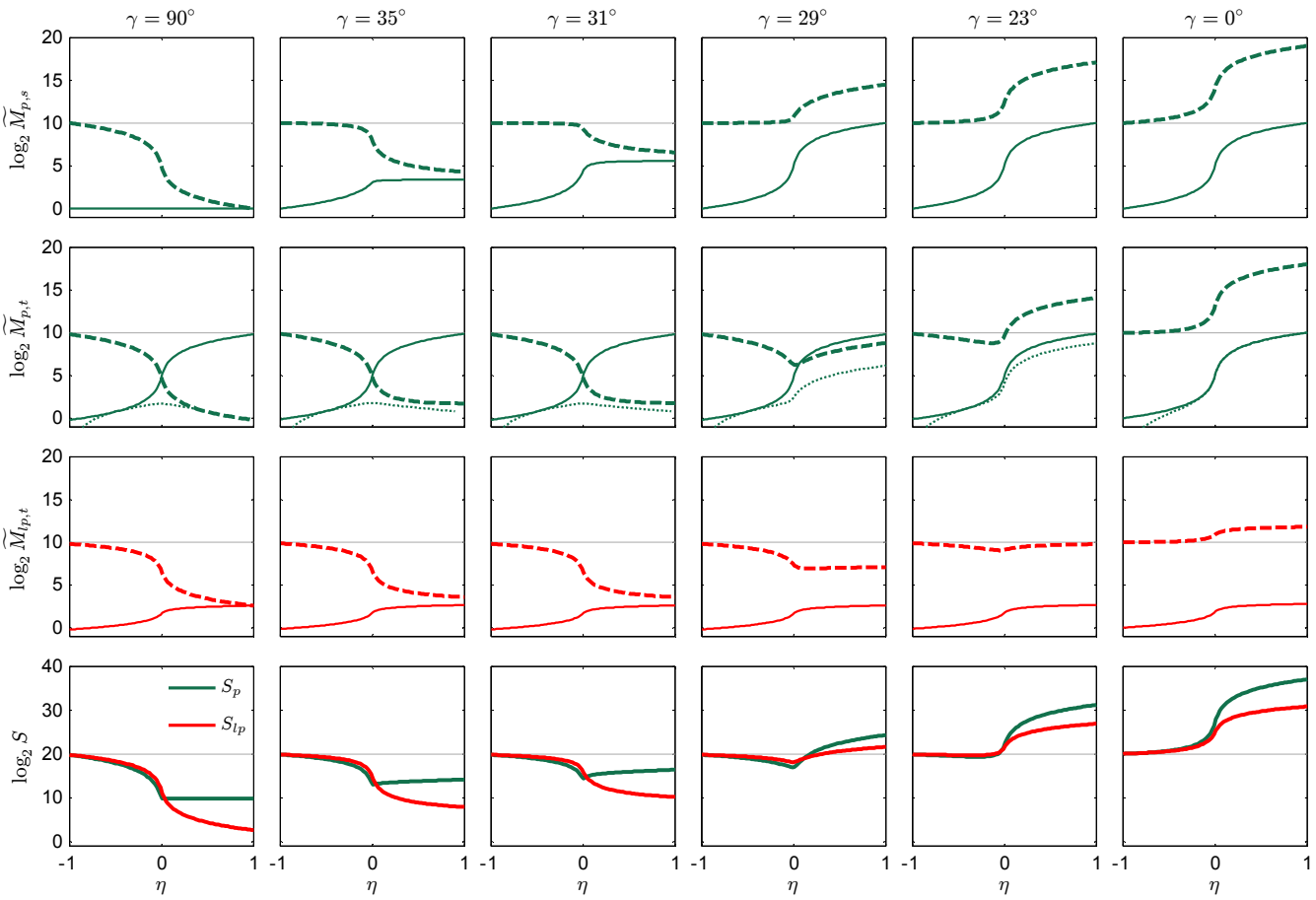

Figure 29: Perspective error over all warping parameters for several light directions. *These graphs plot the error at $v = v_{n1}$ (heavy dotted line), $v = 1$ (thin solid line), and for $\widetilde{M}_{p,t}$ $v = v_{\min}$ where $v_{\min}$ is the location of the minimum value. S_p is the same over $\eta \in [0, 1]$ for $\gamma = 90°$. The frustum parameters are $n = 1$, $f = 1024$, and $\theta = 30°$.*

γ_c controls how quickly the transition from low to high error occurs as $\gamma \to 0$. We choose $\gamma_c = \theta + 0.3(90 - \theta)$ so that the error is kept low for $\gamma > \theta$ while extending the transition period. η_c should be 1 for optimal S_{lp} and 0 for optimal S_p. η_b should be 0 to replicate the behavior of the S-optimal curve. However, this causes the value of $\widetilde{M}_{p,s}$ at $v = 1$ to rise to the same value as at $v = v_{n1}$. By decreasing η_b a small amount it is possible to mitigate this effect without affecting S too much. By experimentation we arrive at the value of $\eta_b = -0.2$. The behavior of S using these parameters to choose the warping parameter is fairly consistent over varying θ and f_e/n_e. We have observed that for some light positions not in the yz plane that the warping can be slightly stronger than necessary.

Our shaping function is based on the optimal parameter using S as an error metric. S is only one possible error metric and may not be suited for all applications. Our shaping function may not be able to provide a good fit for other metrics. For this reason we would like to create a more flexible system that would allow a user to build a shaping function interactively using something like smooth splines. By presenting users with graphs similar to those shown in this paper, they could interactively generate a shaping function that meets their particular needs.

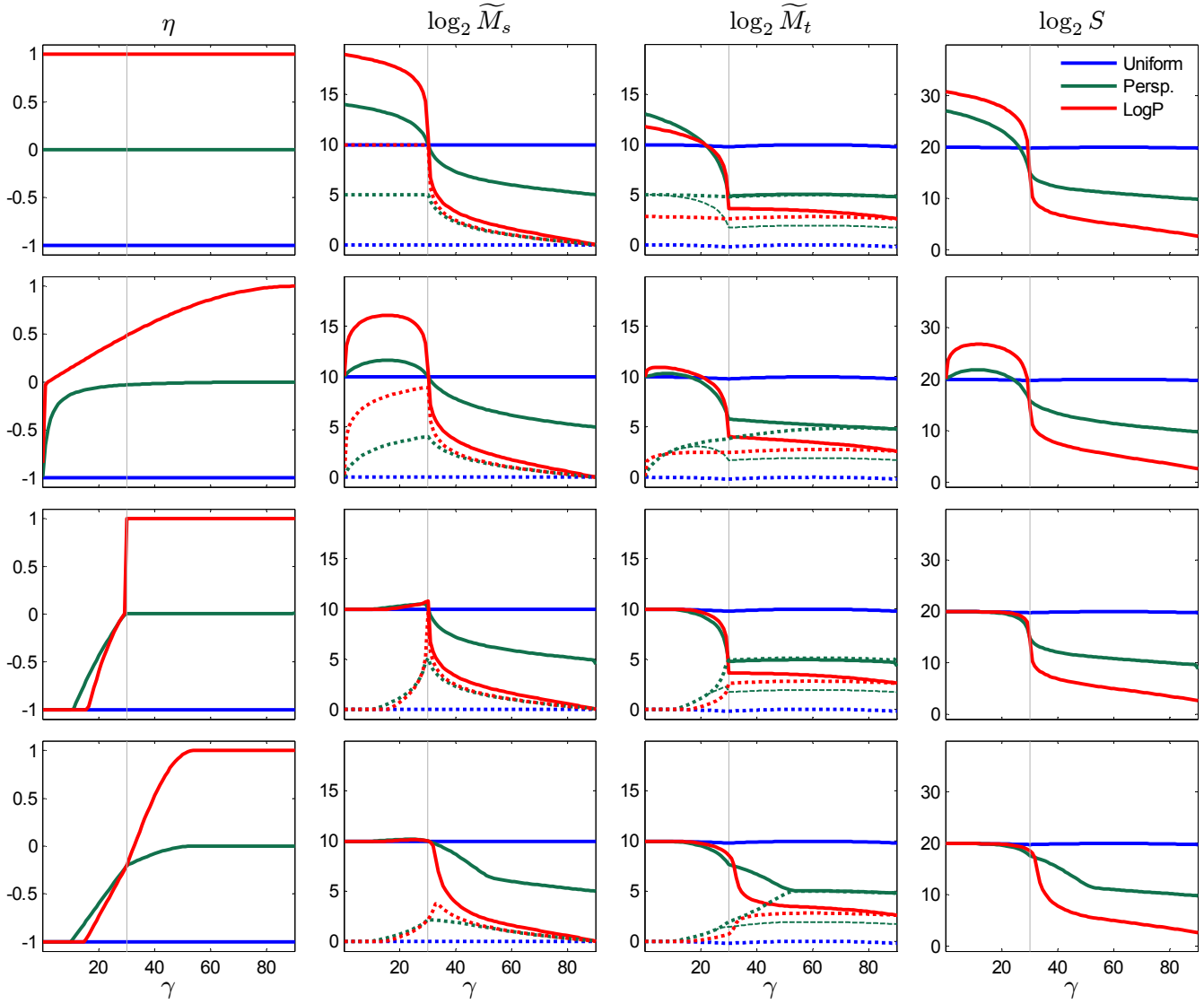

Figure 30: Various shaping functions for the warping parameter. *For the $\widetilde{M}$ graphs, the solid lines are $v = v_n$, dotted lines are $v = 1$, and the dashed line is $v = v_{\min}$ for $\widetilde{M}_{p,t}$. (First row) No shaping function is used to modulate the warping parameter that is optimal for $\gamma = 90°$. The error is extremely high for $\gamma < \theta$. (Second row) The LiSPSM $1/\sin\gamma$ is used to ramp off n'. (n' is converted to η in this graph). (Third row) The warping parameter is chosen to minimize S, but the transition from low to high error at $\gamma = \theta$ is too rapid. (Fourth row) Our shaping function keeps the error low for $\gamma > \theta$ while avoiding excessive error for $\gamma < \theta$. In addition it provides a smoother transition. Note that the uniform parameterization actually does not have a parameter but the other two parameterizations converge to uniform as $\eta \to -1$. The frustum parameters are $n = 1$, $f = 1024$, and $\theta = 30°$.*