Separating Plane Perspective Shadow Mapping

Morten Mikkelsen
mm@ioi.dk

March 29, 2007

Abstract.
This paper proposes a real-time hybrid solution between standard shadow mapping and any of the current perspective shadow mapping based methods. Existing methods are known to have undersampling problems. At little extra cost we combine two methods to minimize undersampling errors. This is done by computing a plane equation which separates the two resulting shadow maps to get the best possible area-based sampling density. An additional property of this separation plane is that objects entirely on one side of the plane only need to be rendered into one map.

1 Introduction

Fast high quality shadows have been in demand for a long time and will most likely continue to be so in the foreseeable future. The most popular gpu based methods with real-time potential today are shadow volumes by Crow [Crow77] and shadow mapping by Williams [Will78]. Shadow volumes have the advantage over shadow mapping that they do not suffer from aliasing problems. In spite of this many developers are discouraged from using shadow volumes because they do not offer the same degree of flexibility as shadow maps: It is impractical to maintain a complete silhouette list in every frame and they are fill-rate intense. This has led to much work over the years on reducing aliasing introduced by shadow mapping. One of the most famous methods to improve on the result is percentage closer filtering [Reev87] which blurs the result by sampling many times in the shadow map within some region around the sampling point. However, this does not diminish the actual undersampling, it only filters by interpolating the binary result of the depth comparison. In recent years additional work on the undersampling problem has been made by [Donn06], [Eric04], [ffbg01] and [aaron05]. This paper aims to reduce the same problem by combining a different branch of methods which will be discussed in the next section.

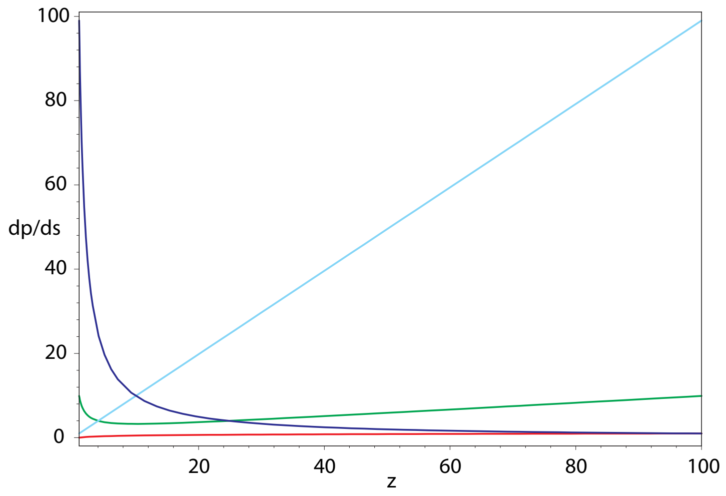


Figure 1: These graphs describe the undersampling from $near$ to far $[1; 100]$ of the camera for four cases. The plots and the term $\frac{dp}{ds}$ are from the analysis of [Wimm04]. The red curve is the result of the ideal logarithmic function. The blue curve represents standard shadow mapping, i.e., pull-back to infinity of the camera projection reference point. Finally the turquoise curve represents applying the camera projection with no additional pull-back and the green curve represents a pull-back of $\sqrt{near \cdot far}$ which results in leveling the extremas at the near and the far plane as explained by [Wimm04].

2 Previous Work

Recently, three primary techniques have been introduced: **psm** [Stamm02], **tsm** [Mart04] and **lispsm** [Wimm04] which all aim to reduce aliasing artifacts when rendering shadows using shadow mapping. They do this by applying a double projection, as opposed to the traditional single projection for the light's view used in standard shadow mapping (**ssm**) [Will78].

The **psm** method applies a projection for the camera first and then the light. The methods **tsm** and **lispsm** on the other hand both apply a projection for the light source first and then apply a projection along the camera's line of sight, projected into the view plane of the light. This way the second projection direction is always perpendicular to the light's view direction. For all three methods the intention is to increase the quantity of pixels used in the shadow map near the location of the camera eye point.

It is shown in [Wimm04] that the ideal mapping into shadow map space is in fact logarithmic. Obviously we cannot achieve this using a linear mapping but it can be approximated using a projection. Perspective shadow mapping based methods (henceforth known as PSM based): **psm**, **tsm** and **lispsm** aim to do so, but as shown by [Wimm04] once a certain depth is reached, the undersampling will continue to rise while for standard shadow mapping it will continue to fall (see figure 1). This paper presents a method that combines PSM based methods with standard shadow mapping to minimize per pixel aliasing for the entire z-range using a separating plane to mark the transition between the two maps. By combining methods, we can keep the undersampling below the green and the blue curves in figure 1. The graph is exact for points of no projection aliasing on the Z-axis of the camera from *near* to far and for a fixed constellation of a camera and light. In section 3 we compare undersampling for the general case.

The analysis of [Wimm04] has inspired [Lloyd05] to reduce perspective aliasing by subdividing the scene and applying a separate warping function to each subdivision. First, the camera's frustum is subdivided according to its visible frustum faces to reduce errors particularly when approaching the dueling frusta case. Second, these frustum faces are further subdivided and warping (as in [Wimm04]) is applied to each subdivision to reduce perspective aliasing. While the method does reduce aliasing it comes at a cost due to the separate rendering of each subdivision, and the use of a complex fragment program for lookup into the subdivided shadow maps. Surprisingly, continous transitions between subdivisions is not adressed by [Lloyd05].

The method presented in this paper divides the shadow mapped scene into two parts and given a chosen metric maintains a continous transition everywhere from one shadow map to the other. It can be used on its own as described here or possibly replace the second subdivision step in [Lloyd].

3 Theory

In this section a few industry terms will be used such as *post-projective space* which is the space in which the view frustum exists as a centered, axis-aligned $2 \times 2 \times 2$ cube ($2 \times 2 \times 1$ for D3D). Additionally texture coordinates will be referred to as either normalized or unnormalized which is not to be confused with vector normalization. Texture coordinates are normalized when a single square unit covers the entire texture and unnormalized when a single square unit only covers a texel. This is similar to the relationship between post-projective space and *screen space* though for a given depth the surface of the screen in post-projective space is four square units and not one.

It will be implied throughout this section that all the applied matrices are nonsingular. Let M_1 and M_2 be two such 4×4 transformations from world space to separate texture spaces of unnormalized texture coordinates. In this section a simple inequality will be derived for which of the two provide better area sampling density at a given point.

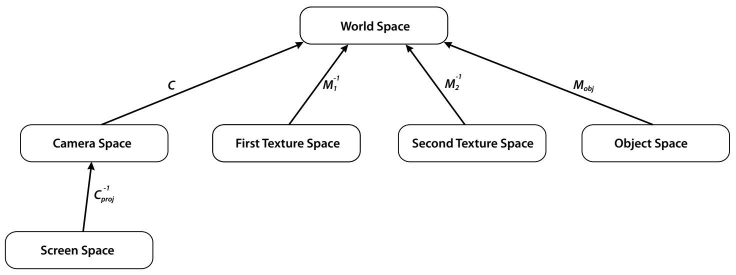


Figure 2: This shows the transformation hierarchy described in this section.

3.1 A general comparison test

Let C be the camera to world matrix, C_{proj} is the camera projection matrix and (s, t, q) is a point in screen space given some arbitrary range for the depth buffer $[z_{min}; z_{max}]$ (see figure 2).

$$M_i' = M_i \cdot C \cdot C_{proj}^{-1}, \qquad i \in \{1, 2\}$$

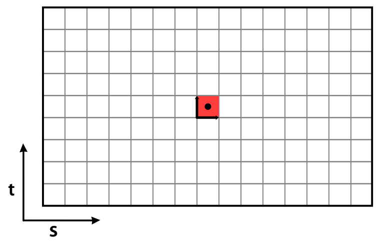
$$\begin{pmatrix} x_i(s,t,q) \\ y_i(s,t,q) \\ z_i(s,t,q) \\ w_i(s,t,q) \end{pmatrix} = M_i' \cdot \begin{pmatrix} s \\ t \\ q \\ 1 \end{pmatrix} \tag{1}$$

The final texture coordinates are computed using the following equations:

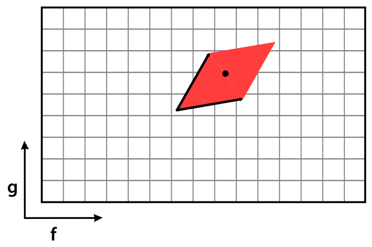
$$\begin{aligned} f_i(s,t,q) &= x_i(s,t,q)/w_i(s,t,q) \\ g_i(s,t,q) &= y_i(s,t,q)/w_i(s,t,q) \end{aligned}$$

The signed area covered by a screen space pixel in the texture maps can be evaluated by the determinant of the Jacobian matrix of (f_i, g_i) with respect to s and t (see figure 3).

$$\begin{aligned} J(f_i, g_i) &= \begin{bmatrix} \frac{df_i}{ds} & \frac{df_i}{dt} \\ \frac{dg_i}{ds} & \frac{dg_i}{dt} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\frac{dx_i}{ds} \cdot w_i - x_i \cdot \frac{w_i}{ds}}{w_i^2} & \frac{\frac{dx_i}{dt} \cdot w_i - x_i \cdot \frac{w_i}{dt}}{w_i^2} \\ \frac{\frac{dy_i}{ds} \cdot w_i - y_i \cdot \frac{w_i}{ds}}{w_i^2} & \frac{\frac{dy_i}{dt} \cdot w_i - y_i \cdot \frac{w_i}{dt}}{w_i^2} \end{bmatrix} \end{aligned}$$



(a) screen space.



(b) texture space.

Figure 3: Subfigure 3(a) shows the area of a single pixel in screen space. Subfigure 3(b) illustrates the pixel after a transformation and projection into texture space, this transformation also depends on the depth of the pixel in screen space.

$$\begin{aligned} det[J(f_i,g_i)] &= \frac{df_i}{ds}\cdot\frac{dg_i}{dt} - \frac{dg_i}{ds}\cdot\frac{df_i}{dt} \\ &= \frac{\left(\frac{dy_i}{ds}\frac{dw_i}{dt} - \frac{dw_i}{ds}\frac{dy_i}{dt}\right)\cdot x_i + \left(\frac{dw_i}{ds}\frac{dx_i}{dt} - \frac{dx_i}{ds}\frac{dw_i}{dt}\right)\cdot y_i + \left(\frac{dx_i}{ds}\frac{dy_i}{dt} - \frac{dy_i}{ds}\frac{dx_i}{dt}\right)\cdot z_i}{w_i^3} \end{aligned}$$

We can simplify this equation a little by introducing the vector

$$\overrightarrow{n_i} = \begin{pmatrix} \frac{dx_i}{ds} \\ \frac{dy_i}{ds} \\ \frac{dw_i}{ds} \end{pmatrix} \times \begin{pmatrix} \frac{dx_i}{dt} \\ \frac{dy_i}{dt} \\ \frac{dw_i}{dt} \end{pmatrix} \tag{2}$$

Let the symbol $\bullet$ denote the dot product between two vectors. Now the signed area can be expressed as

$$det[J(f_i,g_i)] = \frac{\overrightarrow{n_i}\bullet(x_i,y_i,w_i)}{w_i^3}$$

Note that from (1) it follows that $\frac{d}{ds}(x_i,y_i,z_i,w_i)$ and $\frac{d}{dt}(x_i,y_i,z_i,w_i)$ are equal to the first and second column of M_i'.
It now follows that the test for largest sampling density can be done as:

$$\begin{aligned} |det[J(f_1,g_1)]| &> |det[J(f_2,g_2)]| \\ &\Leftrightarrow \\ |\overrightarrow{n_1}\bullet(x_1,y_1,w_1)|\cdot|w_2^3| &> |\overrightarrow{n_2}\bullet(x_2,y_2,w_2)|\cdot|w_1^3| \end{aligned} \tag{3}$$

An important subtlety to acknowledge is (3) takes points of the form $(s,t,q,1)^T$ as input. However the inequality will work for a point even when the last component is unequal to 1 since the effect of using it on such a point $(s',t',q',r')^T$

corresponds to applying the inequality for $(s'/r', t'/r', q'/r', 1)^T$ and scaling by r'^4 on both sides which is redundant.
This means given a transformation M_{obj} from some arbitrary object space to world space, we can transform and test points from this object space without having to make a stop in screen space to perform the divide. We do this by using the transformation sequence

$$M'_i \cdot C_{proj} \cdot C^{-1} \cdot M_{obj} = M_i \cdot M_{obj}$$

The vector $\overrightarrow{n_i}$ is still derived from M'_i and is constant. Now let us re-examine $M' = M'_1$ (the i will be omitted for now). We have $M' = M \cdot C \cdot C^{-1}_{proj}$ and let

$$A = M \cdot C = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

$$C^{-1}_{proj} = \begin{bmatrix} s'_x & 0 & 0 & c'_x \\ 0 & s'_y & 0 & c'_y \\ 0 & 0 & 0 & -1 \\ 0 & 0 & k'_1 & k'_2 \end{bmatrix}$$

$$M' = A \cdot C^{-1}_{proj} = \begin{bmatrix} s'_x a & s'_y b & k'_1 d & c'_x a + c'_y b - c + k'_2 d \\ s'_x e & s'_y f & k'_1 h & c'_x e + c'_y f - g + k'_2 h \\ s'_x i & s'_y j & k'_1 l & c'_x i + c'_y j - k + k'_2 l \\ s'_x m & s'_y n & k'_1 p & c'_x m + c'_y n - o + k'_2 p \end{bmatrix}$$

Given (2) we now have

$$\overrightarrow{n} = \overrightarrow{n_1} = s'_x \cdot s'_y \cdot \begin{pmatrix} a \\ e \\ m \end{pmatrix} \times \begin{pmatrix} b \\ f \\ n \end{pmatrix}$$

and after evaluation of the first term in (3) we get

$$\begin{aligned} |\overrightarrow{n} \bullet (x, y, w)| &= \left| \begin{bmatrix} \overrightarrow{n}_x & \overrightarrow{n}_y & 0 & \overrightarrow{n}_z \end{bmatrix} \cdot M' \cdot \begin{bmatrix} s \\ t \\ q \\ 1 \end{bmatrix} \right| \\ &= |s'_x s'_y \cdot (det(A_{33}) \cdot (k'_1 q + k'_2) - det(A_{34}))| \qquad (4) \end{aligned}$$

The syntax A_{ij} represents the 3×3 submatrix we obtain by removing the ith row and the jth column. Thus it is clear the term only depends on q.

In the following section 3.2 the analysis will be focused on shadow mapping.

3.2 Shadow Mapping

In this section *shadow map space* will be analogous to texture map space (or texture space) from section 3.1.

For a given constellation of a camera and a light we can evaluate a pair of projection matrices M_i and M_j from world space to the shadow map ($i, j \in \{1, 2, 3, 4, 5\}$ and $i \neq j$).

1. ***ssm***, standard full light frustum shadow mapping
2. ***fsm***, also standard but focused on the intersection between bounds.
3. ***tsm***, trapezoidal shadow mapping
4. ***lispsm***, light space perspective shadow mapping
5. ***psm***, perspective shadow mapping

Let I be the intersection volume between the world bound, camera frustum and light frustum. Generally for methods 2-5, the side-planes and back-plane of the light frustum are placed tightly around I. Subsequently the near plane is pushed forward until it reaches the intersection volume between this new light frustum and the world bound. This completes the new light frustum, let V denote its volume (see figure 4). This process is what distinguishes **fsm** from **ssm**.

Theorem

For any such pair M_i and M_j, (3) is reduced to the point set on one side of a plane equation. Furthermore the resulting plane contains the eye point of the light.

Proof

This will be shown through examination of $det(A_{33})$ and $det(A_{34})$ for each of the 5 methods. The proof will be divided into two parts, the first part will prove the theorem for the first four methods and the second part will cover the last of the five methods (**psm**).

Part 1:
The first four methods have roughly the same transformation sequence.

$$A = N_t \cdot L_{proj} \cdot B$$

The matrix B is the camera to light matrix and note that our only assumption about B is that it has an inverse and that the bottom row is $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$.

$$B = \begin{bmatrix} \overrightarrow{c_1}_x & \overrightarrow{c_2}_x & \overrightarrow{c_3}_x & t_x \\ \overrightarrow{c_1}_y & \overrightarrow{c_2}_y & \overrightarrow{c_3}_y & t_y \\ \overrightarrow{c_1}_z & \overrightarrow{c_2}_z & \overrightarrow{c_3}_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

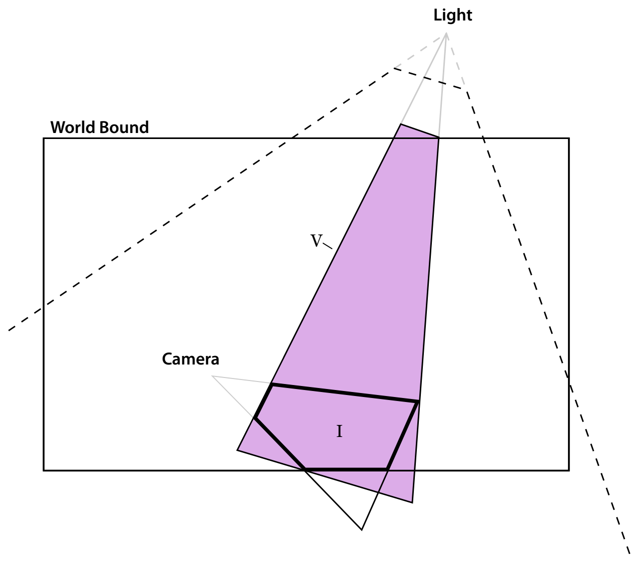


Figure 4: The process of focusing a light on the intersection volume I between bounds. The new light volume V is shown here in purple.

The matrix L_{proj} is the projection matrix into the light's post-projective space.

$$L_{proj} = \begin{bmatrix} r_x & 0 & l_x & 0 \\ 0 & r_y & l_y & 0 \\ 0 & 0 & o_1 & o_2 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

The matrix N_t is a sequence of transformations which are designed to transform a trapezoidal bound around I (possibly a sheared one depending on the implementation) into a nice quadratic square as seen from the light. This principle is shared between **tsm** and **lispsm**. For **ssm** and **fsm** we will simply consider the matrix N_t the identity matrix.
The steps involved in creating the transformation N_t are outlined in [Mart04] and will not be explained here in detail.

It will now be shown that the matrix N_t is of the form

$$N_t = \begin{bmatrix} k_{11} & k_{12} & 0 & k_{14} \\ k_{21} & k_{22} & 0 & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & 0 & k_{44} \end{bmatrix}$$

Notice that the product of two such matrices results in a matrix of the same form. It just so happens that every matrix in the sequence of N_t is of this form.

$$N_t = T' \cdot S \cdot N_{proj} \cdot T \cdot R$$

The matrices T' and T are translations and S is a nonuniform scale. The matrices on the left side T' and S are not part of the original definition of N_t but have been added here to complete the transformation into unnormalized coordinates. The matrix R is a rotation (2D) around the Z-axis and N_{proj} is a projection along the Y-axis.

$$N_{proj} = \begin{bmatrix} q_{11} & q_{12} & 0 & 0 \\ 0 & q_{22} & 0 & q_{24} \\ 0 & q_{32} & q_{33} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

So they all comply and thus N_t is compliant with the form. In the case of **fsm** or **ssm**, N_t is the identity which is also compliant. We can now take advantage of the three zeros in the third column of N_t, fourth column of L_{proj} and the fourth row of B.

$$\begin{aligned} A_{33} &= N_{t33} \cdot L_{proj34} \cdot B_{43} \Rightarrow det(A_{33}) = det(N_{t33}) \cdot det(L_{proj34}) \cdot det(B_{43}) \\ A_{34} &= N_{t33} \cdot L_{proj34} \cdot B_{44} \Rightarrow det(A_{34}) = det(N_{t33}) \cdot det(L_{proj34}) \cdot det(B_{44}) \end{aligned}$$

Note that since B is the camera to light matrix we can obtain the position of the light in camera space using $B^{-1} \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ which after division by $w = 1$ results in $\ell = B_{44}^{-1} \begin{bmatrix} -t_x & -t_y & -t_z \end{bmatrix}^T$. It now follows that:

$$\ell_z = \left(B_{44}^{-1} \begin{bmatrix} -t_x \\ -t_y \\ -t_z \end{bmatrix} \right)_z = \frac{-(\overrightarrow{c_1} \times \overrightarrow{c_2})^T}{det(B_{44})} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \frac{-det\,(B_{43})}{det\,(B_{44})}$$

And now we have $det(A_{33}) = -\ell_z \cdot det(A_{34})$ which by substitution will simplify (4) further.

$$\overrightarrow{n} \bullet (x, y, w) = -s'_x s'_y det(A_{34}) \cdot (\ell_z(k'_1 q + k'_2) + 1) \tag{5}$$

Now let M_1 and M_2 be two projection matrices evaluated by any of the first four methods and let

$$\begin{aligned} A &= M_1 \cdot C \\ A' &= M_2 \cdot C \end{aligned}$$

We can now substitute (5) for A and A' into (3) and obtain the following

$$|det(A_{34})||w_2^3| > |det(A'_{34})||w_1^3|$$

In addition to this we can take advantage of the fact that given any of the five listed shadow mapping methods we have $\forall p \in V : w(p) > 0$. Finally inequality (3) is reduced to

$$|\sqrt[3]{det(A_{34})}|w_2 - |\sqrt[3]{det(A'_{34})}|w_1 > 0 \tag{6}$$

which is the point set entirely on one side of a plane equation. The fact that the plane contains the eye point of the light is a result of the following:

$$L_{proj} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ o_2 \\ 0 \end{bmatrix}$$

So $w = 0$ for **ssm** and **fsm**. Given the known form of N_t this property also holds for **tsm** and **lispsm**.

$$N_t \cdot \begin{bmatrix} 0 \\ 0 \\ o_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ k_{33}o_2 \\ 0 \end{bmatrix}$$

Part 2:
In the second part it will be shown that the proof still holds for the last method. The exact details of the **psm** transformation sequence will not be explained here but the concept of the method is to setup the light projection in post-projective space of the camera, thereby increasing the quantity of shadow map pixels available near the camera eye point.
To allow **psm** to support the pull-back [Kozl04], instead of using the actual projection C_{proj}, an initial translation T', followed by a variant C'_{proj} is used. This allows the frustum to be enlarged which gives you the freedom to tweak the xy-distribution as seen from the light.

For **psm** the transformation sequence M from world space to shadow map space is

$$M = L_{proj} \cdot Q \cdot C'_{proj} \cdot T' \cdot C^{-1}$$

and once again we evaluate A, that is the transformation from camera space to shadow map space.

$$A = M \cdot C = L_{proj} \cdot Q \cdot C'_{proj} \cdot T'$$

The matrix L_{proj} is simply a chosen projection matrix into shadow map space and the remaining matrix Q has the following form:

$$Q = \begin{bmatrix} \overrightarrow{r_1}_x & \overrightarrow{r_1}_y & \overrightarrow{r_1}_z & t'_x \\ \overrightarrow{r_2}_x & \overrightarrow{r_2}_y & \overrightarrow{r_2}_z & t'_y \\ \overrightarrow{r_3}_x & \overrightarrow{r_3}_y & \overrightarrow{r_3}_z & t'_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Compared to the first four methods the transformation sequence is quite different however as it turns out the same strategy, as that of the first part, can be used to complete the proof.

As previously mentioned ℓ, is the eye of the light in camera space. The position $\begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ is the same point transformed into the modified post-projective space of the camera, that is $(C'_{proj} \cdot T') \begin{bmatrix} \ell_x & \ell_y & \ell_z & 1 \end{bmatrix}^T$ (after the divide). And additionally, $\begin{bmatrix} t'_x & t'_y & t'_z \end{bmatrix}^T = Q_{44} \begin{bmatrix} -t_x & -t_y & -t_z \end{bmatrix}^T$ so we can split up Q accordingly.

$$Q = \begin{bmatrix} 1 & 0 & 0 & t'_x \\ 0 & 1 & 0 & t'_y \\ 0 & 0 & 1 & t'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{r_1}_x & \overrightarrow{r_1}_y & \overrightarrow{r_1}_z & 0 \\ \overrightarrow{r_2}_x & \overrightarrow{r_2}_y & \overrightarrow{r_2}_z & 0 \\ \overrightarrow{r_3}_x & \overrightarrow{r_3}_y & \overrightarrow{r_3}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \overrightarrow{r_1}_x & \overrightarrow{r_1}_y & \overrightarrow{r_1}_z & 0 \\ \overrightarrow{r_2}_x & \overrightarrow{r_2}_y & \overrightarrow{r_2}_z & 0 \\ \overrightarrow{r_3}_x & \overrightarrow{r_3}_y & \overrightarrow{r_3}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rows of Q_{44} are typically a reconstructed version of the light's orientation. It is, however, not a necessary limitation for this to work. We only demand that Q_{44} has an inverse (and that $\begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ is as previously described). The matrix L_{proj} is as mentioned a chosen projection matrix for the light with three appropriate zeros in the last column.

$$C'_{proj} = \begin{bmatrix} s_x & 0 & c_x & 0 \\ 0 & s_y & c_y & 0 \\ 0 & 0 & k_1 & k_2 \\ 0 & 0 & -1 & 0 \end{bmatrix} \qquad T' = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$N = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot C'_{proj} \cdot T' = \begin{bmatrix} s_x & 0 & c_x + t_x & s_x p_x + p_z(c_x + t_x) \\ 0 & s_y & c_y + t_y & s_y p_y + p_z(c_y + t_y) \\ 0 & 0 & k_1 + t_z & k_2 + p_z(k_1 + t_z) \\ 0 & 0 & -1 & -p_z \end{bmatrix}$$

We can now rewrite transformation A as

$$A = L_{proj} \cdot \begin{bmatrix} \overrightarrow{r_1}_x & \overrightarrow{r_1}_y & \overrightarrow{r_1}_z & 0 \\ \overrightarrow{r_2}_x & \overrightarrow{r_2}_y & \overrightarrow{r_2}_z & 0 \\ \overrightarrow{r_3}_x & \overrightarrow{r_3}_y & \overrightarrow{r_3}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot N$$

And similar to part 1, the distribution of zeros leads to

$$\begin{aligned} A_{33} &= L_{proj34} \cdot Q_{44} \cdot N_{43} \Rightarrow det(A_{33}) = det(L_{proj34}) \cdot det(Q_{44}) \cdot det(N_{43}) \\ A_{34} &= L_{proj34} \cdot Q_{44} \cdot N_{44} \Rightarrow det(A_{34}) = det(L_{proj34}) \cdot det(Q_{44}) \cdot det(N_{44}) \end{aligned}$$

Now when evaluating $(C'_{proj} \cdot T') \begin{bmatrix} \ell_x & \ell_y & \ell_z & 1 \end{bmatrix}^T$ we get $t_z = (-k_1) + \frac{-k_2}{p_z + \ell_z}$.

This is used for evaluation of $det(N_{43})$ and $det(N_{44})$.

$$\begin{aligned}
det(N_{44}) &= s_x s_y\,(k_1 + t_z) = s_x s_y \left(\frac{-k_2}{p_z + \ell_z}\right)\\
det(N_{43}) &= s_x s_y\,(k_2 + p_z(k_1 + t_z)) = s_x s_y \left(k_2 + p_z\left(\frac{-k_2}{p_z + \ell_z}\right)\right)\\
&= s_x s_y \left(\frac{k_2 \ell_z}{p_z + \ell_z}\right) = -\ell_z \cdot det(N_{44})
\end{aligned}$$

And again, it follows that $det(A_{33}) = -\ell_z \cdot det(A_{34})$ which once again leads to (5) and subsequently (6).
That $w = 0$ for the eye point of the light is clear since $\left(L_{proj}\left[\begin{array}{cccc} 0 & 0 & 0 & 1 \end{array}\right]^T\right)_w =$ 0 and subsequently the proof is complete.

In the following sections 4 and 5 the theory will be put to good use.

4 Comparing undersampling

This section will explain the principle of undersampling comparison for projective textures in general. Let A and B be two independent 3×4 matrices used for projective texturing, both are maps from camera space and into a space of unnormalized texture coordinates. Let C_{proj} be the 4×4 projection into screen space. The transformations from screen space and into texture space become:

$$\begin{aligned}
M'_1 &= A \cdot C_{proj}^{-1}\\
M'_2 &= B \cdot C_{proj}^{-1}
\end{aligned}$$

Let $(s, t, q) \mapsto (x_i, y_i, w_i), \quad i \in \{1, 2\}$ represent for a given input point the resulting homogeneous coordinates in texture space and let $(f_i, g_i) = (x_i/w_i, y_i/w_i)$ be the final texture coordinates. The best sampling density test can be performed using inequality (3) from section 3.1.
The terms x_i, y_i and w_i are all linear maps and since $\vec{n_i}$ is a constant vector, $\vec{n_i} \bullet (x_i, y_i, w_i)$ is a linear map too. This allows us to optimize by evaluating w_i and $\vec{n_i} \bullet (x_i, y_i, w_i)$ at vertex shader level and then do the remainder of the calculation at fragment shader level. Only a single interpolator of 4 floats is required for this.

In section 5 we will use the additional analysis from section 3 on shadow mapping specifically which reduces the comparison test even more.

5 The Separating Plane

This section compares the undersampling between five variants of shadow mapping. Now given one constellation of a camera and a light, let A and B be a pair of 4×4 matrices evaluated by two different methods chosen from any of the five following methods listed here. As in the previous section A and B are maps from camera space.

1. ***ssm***, standard full light frustum shadow mapping
2. ***fsm***, also standard but focused on an intersection between bounds.
3. ***tsm***, trapezoidal shadow mapping
4. ***lispsm***, light space perspective shadow mapping
5. ***psm***, perspective shadow mapping

Again let I and V be given as described in section 3.2 (see figure 4). Now given such A and B it was shown in section 3 that for all points inside V inequality (3) is reduced to the point set on one side of a plane equation.

$$(|\sqrt[3]{det(A_{34})}|B_4 - |\sqrt[3]{det(B_{34})}|A_4) \cdot \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} > 0 \tag{7}$$

Here the syntax A_i denotes the ith row of the matrix A and A_{ij} is the 3x3 submatrix obtained by removing the ith row and the jth column. The coefficients in the first term of inequality (7) are those of a plane equation in camera space (since A and B are transformations from camera space). If so desired the plane can be transformed into some other space of preference using the inverse transposed of the matrix. The vertices of the mesh can be inserted into the plane equation in the vertex shader (or fragment shader) but the actual test of the sign must be done in the fragment shader.

For all vertices inside the intersection between V and the plane, $|det[J(f_1, g_1)]| = |det[J(f_2, g_2)]|$ which means the area sampling density is the same for both maps given such a point. This means for our choice of metric we get a continuous but not necessarily smooth transition from one map to the other.

6 Optimization

The separating plane has an additional property which can be used to advantage for further optimization. According to section 3, the plane contains the eye point of the light. The effect of this is that the plane becomes a separating line in the shadow map, which again means that if two models are on separate sides of the plane, they will have no pixel overlap in the shadow map. This leads to an elegant optimization which allows us to render an object only once if it is entirely on one side of the plane, which will generally be the case.

Assuming we are applying the separating plane technique for **fsm** and for a PSM based method, the plane will separate in such a way that the part of the camera frustum close to the eye point will belong to the PSM based method and

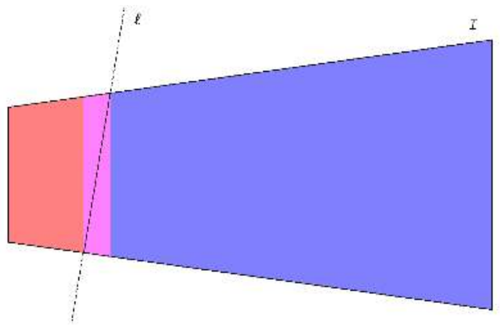


Figure 5: This shows the intersection volume I as seen from the light source. The separation plane in this space appears as the line ℓ. The region of I filled with the color purple is where the two reduced focus regions overlap. This region will appear on both shadow maps.

the part on the other side will always belong to the **fsm**. With this in mind an additional optimization becomes possible: Instead of letting both methods be derived from a focused view on all of I, they can alternatively focus on the part of I which belongs to their side of the plane. Of course each time the focus regions are changed, the transformations have to be re-evaluated and subsequently so does the separation plane. The implementation of this paper does iterations to find a suitable location for the plane and the focus regions. Convergence appears to occur rapidly, 10-20 iterations are used for this implementation. In the last iteration the plane equation is not recalculated, (unless the number of iterations is set to 1) this is to make sure that the intersection between I and the separating plane is entirely contained inside the current overlap between the final two focus regions (see figure 5). This way it is still only necessary to check which side of the plane a pixel/point is on to choose the right map.

7 Results

The method described in this paper was implemented for a point light source and tested on **fsm** and **psm**.

For the implementation of **psm** no read back from the depth buffer is performed (as explained in the original paper [Stamm02]), however, a pull-back value similar to the description in [Wimm04] is used.

Additionally, the problem of a light source behind the camera is solved by using the method described in [Kozl04]. The results of this paper are rendered on an NV7800 into a 1280×720 draw buffer and a 1024×1024 shadow map for **fsm** and **psm** is used.

For the case shown in figure 6 the recorded timings listed in table 1 show that compared to **psm**, the penalty for introducing the separating plane algorithm was roughly a 9% increase in execution time.

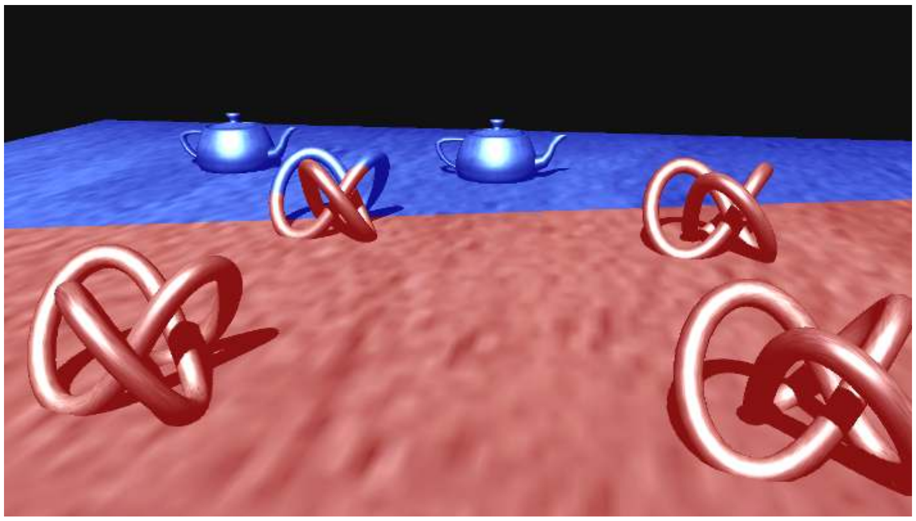


Figure 6: This scene shows two teapots entirely on the **fsm** side of the separating plane, three torus knots entirely on the **psm** side and one torus knot which intersects the plane and thus needs to be rendered into both maps.

Method	**time**
Single shadow map **psm**	2148
psm & **fsm** combined	2340

Table 1: This table shows the execution time in microseconds

The result from **ssm** (non-view-dependent) can be seen in figure 7. To get an idea of the scene setup a mesh of the camera frustum was rendered into picture 7(b) which was taken from the light source.

All pixels in the following figures have either a red or blue tint, red means better sampling density is achieved by sampling from the shadow map generated by **psm** and blue means **fsm** provides a better result.

Figure 8 shows **fsm** and **psm** used separately. Clearly the red area of the picture **psm** provides better results. A zoomed-in version of a blue section of a shadow in the background is provided to show how **fsm** does a better job here.

A combination of the two methods as suggested by this paper can be seen in figure 9. Subfigure 9(a) and 9(c) show the result of both maps focused on all of I, and 9(b) and 9(d) are the result of spending 20 iterations on focusing the maps on I at separate sides of the final separation plane. The iterations are spent on analysis of only three input bounds, so this process is not expensive.

Figure 10 shows the result of the maps as seen from the light source. Although some increase in quality for figure 9 can be seen, it does not appear significant. Because the separating plane is close to the eye point, only a little is cut off the focus region, thus giving only a slight improvement in the **fsm** region. Moreover, since the PSM based method has used up most of its pixels by the time it reaches the separation plane, only little is gained by cutting off from the focus region the part of I on the **fsm** side. Whether or not this optimization is actually negligible depends on the size of I and the pull-back.

8 Conclusions

Given a focused standard shadow map and a perspective shadow mapping based map, this paper presents a simple way to compute an exact separation plane which divides the volume of the light into two parts. One part achieves better area sampling density everywhere using the focused standard shadow map. The second part achieves better results using the perspective shadow map. Using this combination of shadow maps we achieved a significant improvement in shadow quality, at little extra cost in performance.

The method introduces no new singularities but does to some extent inherit the limitations known from the chosen PSM based method. If poor area sampling density is found in both maps then as a consequence the quality of the shadow will be poor. The approach provides only a method to make the better choice at relatively low cost.

All vertices inside the intersection between the plane and the volume of the light will achieve the same area sampling density in either map. This results in a continuous (but not necessarily smooth) transition from one map to the other.

Furthermore, we have shown that this technique can be optimized so only meshes which intersect the separation plane need to be rendered into both maps. This allows most meshes inside the light's volume to be rendered into just one of the two maps.

As an additional optimization one can focus the two maps on separate sides of the separation plane which provides a slight but possibly negligible increase in quality of results.

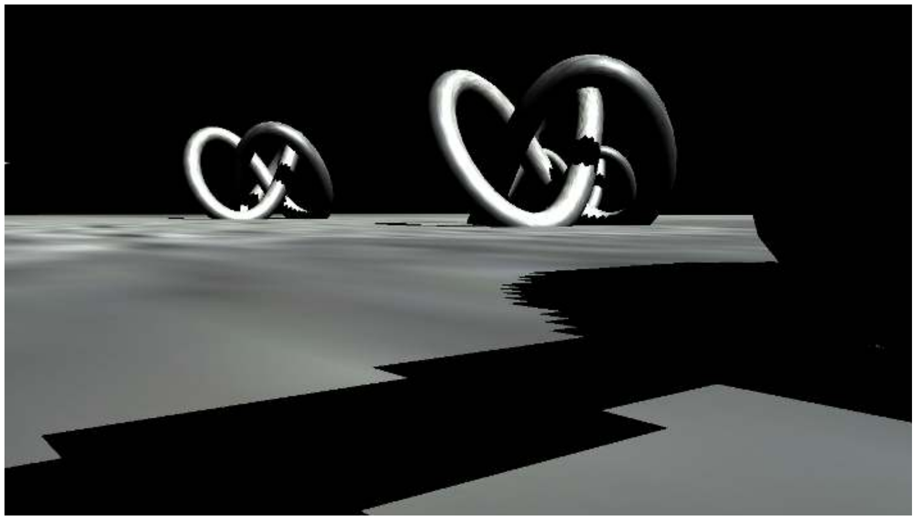
Acknowledgments.
This author would like to thank Michael Wimmer for helpful discussions, and Kasper H. Nielsen and Trine Mikkelsen for constructive comments and proof reading.

References

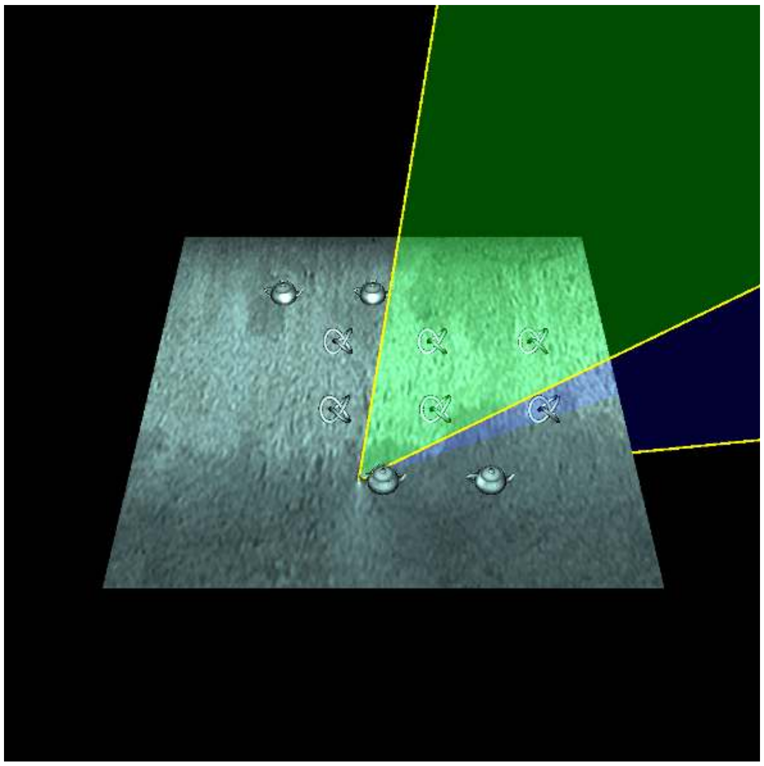
[Mart04] Martin, T., Tan, Tiow-Seng: "Anti-aliasing and Continuity with Trapezoidal Shadow Maps", Eurographics Symposium on Rendering, pp. 153–160, 2004.

[Wimm04] Michael Wimmer, Daniel Scherzer, Werner Purgathofer: "Light Space Perspective Shadow Maps", orignally published at Eurographics Symposium on Rendering, revised version June 10, 2005.

[Stamm02] Marc Stamminger, George Drettakis: "Perspective Shadow Maps", Computer Graphics, SIGGRAPH 2002 Proceedings, pp. 557–562, 2002.

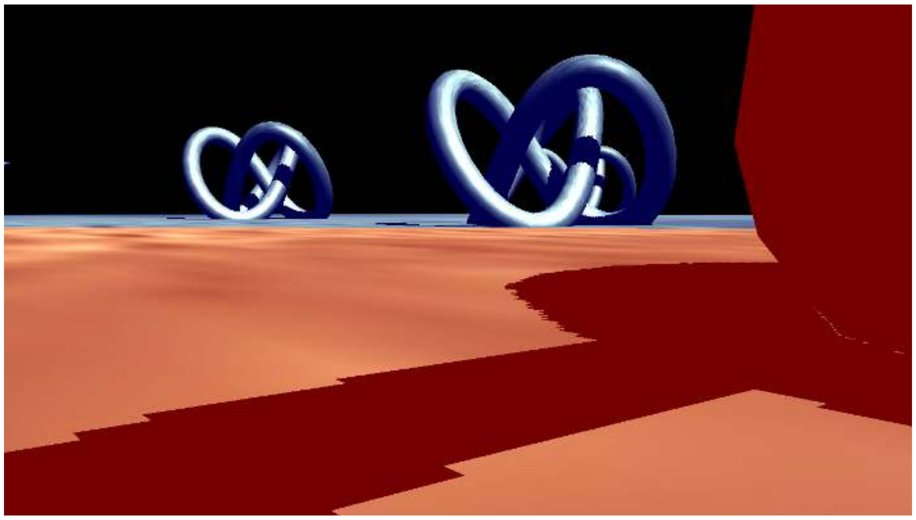


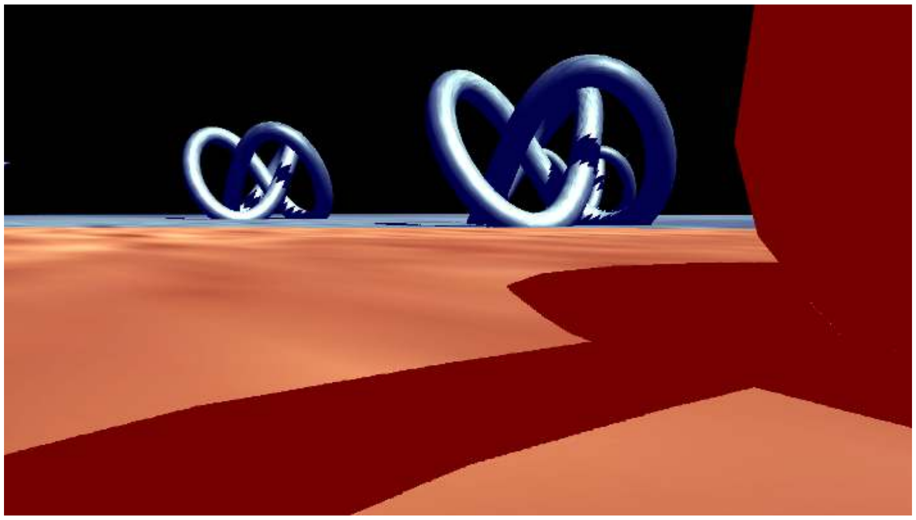
(a) ssm

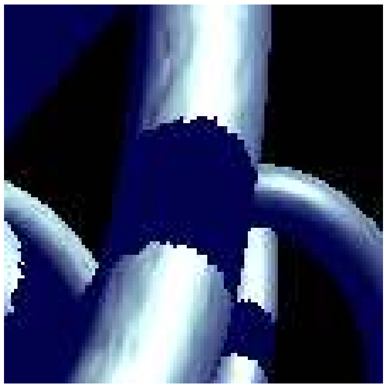


(b) seen from the light source with superimposed camera volume

Figure 7: This shows the result of a non-view-dependent standard shadow mapping.


(a) fsm


(b) psm


(c) zoom in on fsm

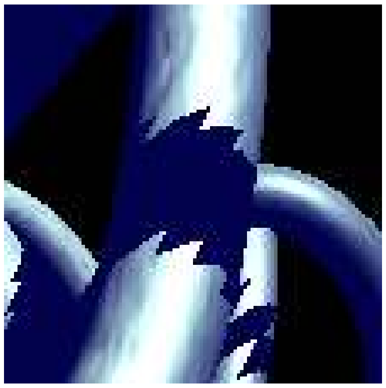
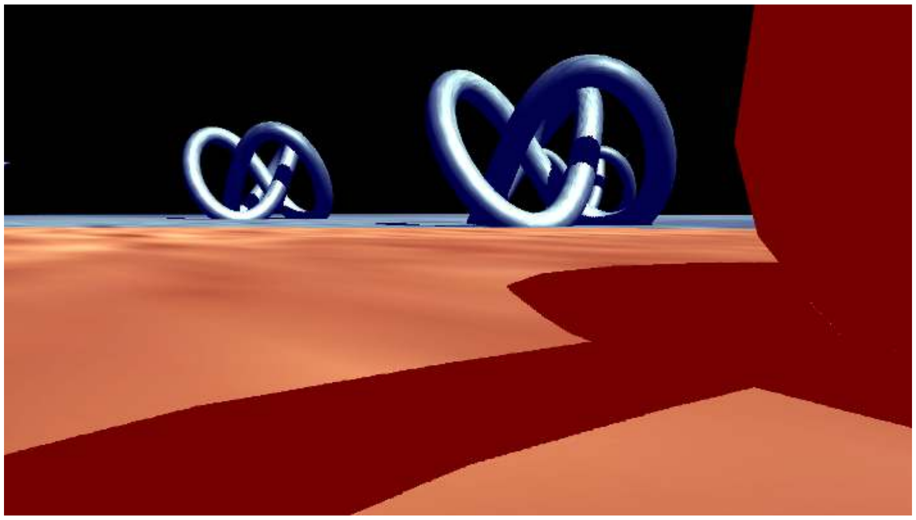
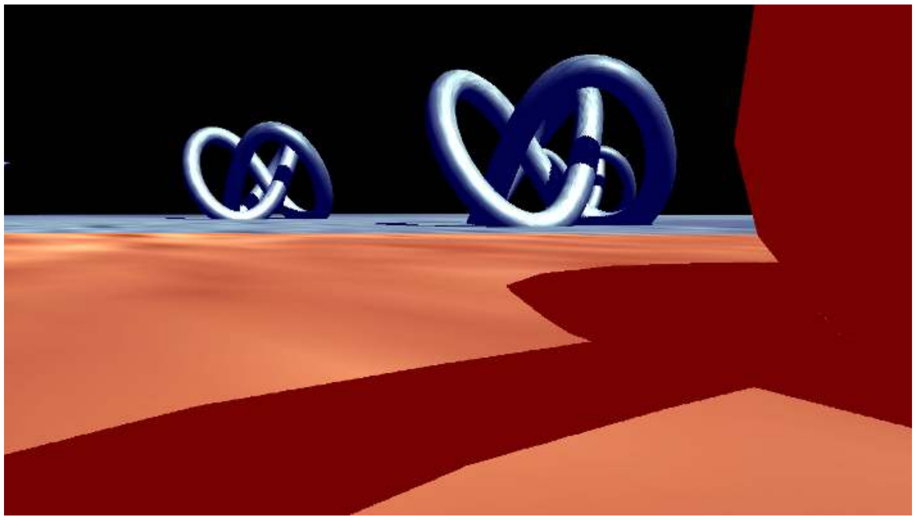
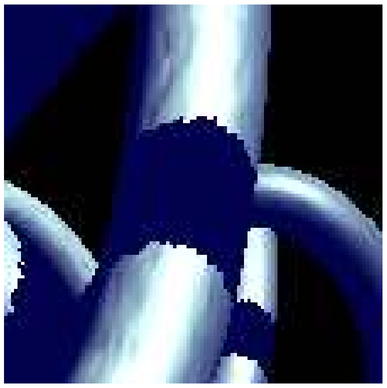

(d) zoom in on psm

Figure 8: This shows the result of **fsm** and **psm** used separately. **psm** provides better results for the pixels (tinted in red) and **fsm** works better for things further away (tinted in blue). Tinting is done by using the separation plane test.


(a) combined, 1 iteration.


(b) combined, 20 iterations.


(c) zoom in on combined, 1 iteration.

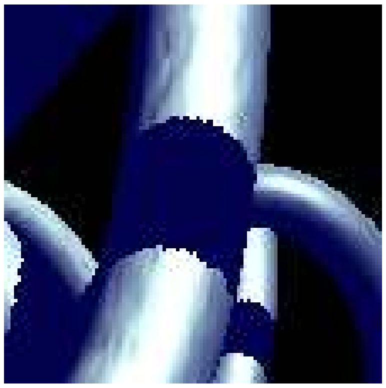
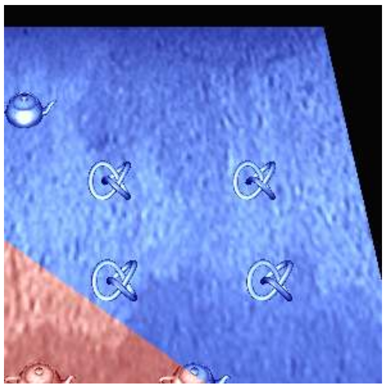
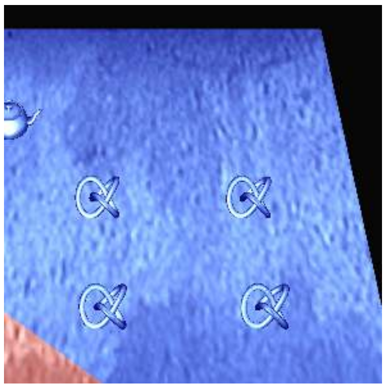

(d) zoom in on combined, 20 iterations.

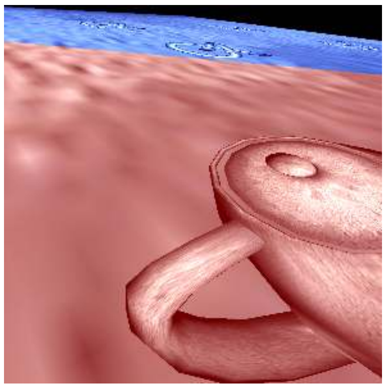
Figure 9: This shows **fsm** and **psm** used together. Subfigure 9(a) shows the result of both shadow maps focused on all of the intersection volume I. Subfigure 9(b) shows the result of focusing the maps on I at separate sides of the separation plane. Subfigures 9(c) and 9(d) are blow ups of 9(a) and 9(b) respectively.



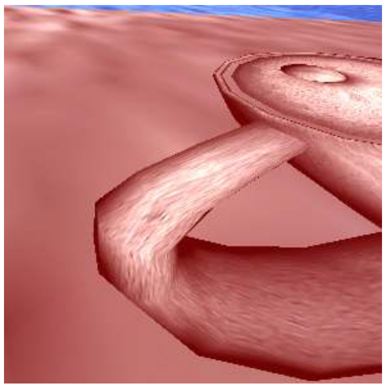
(a) fsm, 1 iteration.



(b) fsm, 20 iterations.



(c) psm, 1 iteration.



(d) psm, 20 iterations.

Figure 10: Here we see the results of **fsm** and **psm** as seen from the light source. As shown here 10(b) and 10(d) are a bit more focused due to the additional iterations.

[Lloyd05] Brandon Lloyd, Sung-eui Yoon, David Tuft, Dinesh Manocha: ”Subdivided Shadow Maps”, Technical Report TR05-024, University of North Carolina at Chapel Hill, 2005.

[Will78] Lance Williams: ”Casting curved shadows on curved surfaces”, Proceedings of the 5th annual conference on Computer graphics and interactive techniques, p. 270–274

[Reev87] William T. Reeves, David H. Salesin, Robert L. Cook: ”Rendering antialiased shadows with depth maps”, Proceedings of the 14th annual conference on Computer graphics and interactive techniques, p. 283–291

[Crow77] Franklin C. Crow: ”Shadow algorithms for computer graphics”, Proceedings of the 4th annual conference on Computer graphics and interactive techniques, p. 242–248

[Kozl04] Simon Kozlov: ”Perspective Shadow Maps: Care and Feeding”, GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics, p. 217–244

[Donn06] William Donnelly, Andrew Lauritzen: ”Variance Shadow Maps”, Proceedings of the 2006 symposium on Interactive 3D graphics and games, p. 161 - 165

[Eric04] Eric Chan, Fredo Durand: ”An Efficient Hybrid Shadow Rendering Algorithm”, Eurographics Symposium on Rendering, 2004.

[ffbg01] Randima Fernando, Sebastian Fernandez, Kavita Bala, Donald P. Greenberg: ”Adaptive Shadow Maps”, Computer Graphics, SIGGRAPH 2001 Proceedings.

[aaron05] Aaron Lefohn, Shubhabrata Sengupta, Joe Kniss, Robert Strzodka, John D. Owens: ”Dynamic Adaptive Shadow Maps on Graphics Hardware”, Technical report, 2005.