Eurographics Symposium on Rendering (2004)
A. Keller, H. W. Jensen (Editors)

Anti-aliasing and Continuity with Trapezoidal Shadow Maps

Tobias Martin and Tiow-Seng Tan

School of Computing, National University of Singapore

Abstract

This paper proposes a new shadow map technique termed trapezoidal shadow maps to calculate high quality shadows in real-time applications. To address the resolution problem of the standard shadow map approach, our technique approximates the eye's frustum as seen from the light with a trapezoid to warp it onto a shadow map. Such a trapezoidal approximation, which may first seem straightforward, is carefully designed to achieve the goal of good shadow quality for objects from near to far, and to address the continuity problem that is found in all existing shadow map approaches. The continuity problem occurs mainly when the shadow map quality changes significantly from frame to frame due to the motion of the eye or the light. This results in flickering of shadows. On the whole, our proposed approach is simple to implement without using complex data structures and it maps well to graphics hardware as shown in our experiments with large virtual scenes of hundreds of thousands to over a million of triangles.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation–Anti-aliasing, bitmap and framebuffer operations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism–Color, shading, shadowing, and texture

1. Introduction

Real time shadow generation has been gaining a lot of attention recently due to the growing support of programmable graphics processing units in PCs and game consoles. In many applications, shadows are important because they add further realism to scenes and provide additional depth cues. Finding ways to calculate shadows started a few decades ago; see the survey by [WPF90]. We note that in most of these techniques, there is a tradeoff between shadow quality and rendering time. Recent approaches, including our work here, are based on the standard shadow map algorithm originally proposed in [Wil78]. This two-pass algorithm is neat and easy to understand. In the first pass, the scene is rendered from the viewpoint of the light with depth buffer enabled. This buffer is then stored into an image called shadow map. In the second pass, the scene is rendered from the viewpoint of the eye incorporating shadow determination for each pixel. A pixel is in shadow if the z-value of its corresponding surface point when transformed into the light's view is greater than its corresponding depth value stored in the shadow map.

The standard shadow map algorithm is easy to implement and is also fast in its calculation. It is well-suited for all kinds of geometric primitives as well as complex and curved objects. Additionally, its operations can be mapped well to graphics hardware where the depth buffer and projective texture mapping are used to calculate shadows [SKvW*92].

On the other hand, the approach has two well-known drawbacks. Its first drawback is the *resolution problem* where it works well when the light is close to the scene and to the viewpoint of the eye, but produces aliases around shadow boundaries when the light is far away. This is caused by low shadow map resolution (i.e. an under-sampling) in areas where a higher resolution is needed. Besides the practical scenario where only a small texture is available to capture the shadow map, this problem can also arise when the visible region of the eye's frustum occupies only a small fraction in the shadow map. There are approaches that address this issue (see [BAS02], [FFBG01], [SD02] as discussed later in Section 2). However, for real-time interactive applications (with little or no restriction on the motion of the eye, lights and objects), there is no satisfactory shadow map approach that addresses the resolution problem while also bearing in mind the polygon offset problem and the continuity problem discussed subsequently.

Its second drawback is the *polygon offset problem*. Due to

the image-based nature, shadow determination is performed with finite precision which can cause surface acne effects. This can be addressed by finding an offset which is added to the depth values of the shadow map to move the z-values and hence the shadows slightly away from the light [Wil78]. Additionally, in [WM94], the depth values of surfaces that are second nearest to the light source are sampled to address this problem.

The first contribution of this paper is a new approach of calculating shadows using trapezoidal shadow maps which are derived from *trapezoidal approximations* of the eye's frusta as seen from the light. It addresses the resolution problem of the standard shadow map approach to result in enhanced shadow map resolution for both static and dynamic objects from near to far and with no constraint on the relative positions and motions of the eye and the light. At the same time, it does not worsen the polygon offset problem as it uses the programmable graphics pipeline to confine the problem to be as manageable as that in standard shadow maps. The approach is efficient as only the eight corners of the eye's frustum plus the centers of the near and the far plane, rather than the scene, are needed to compute a good trapezoidal approximation. It thus scales well to large scenes. Figure 1 shows an example of our approach.

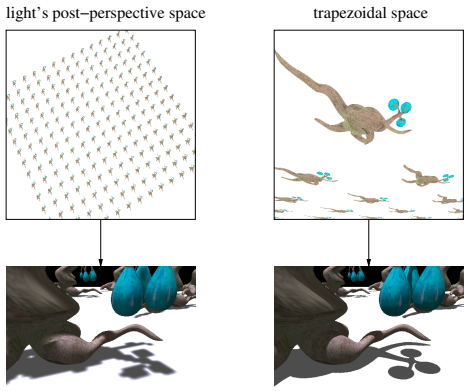

Figure 1: *(a) The shadow map of the scene with 225 regularly spaced plant models is computed directly from the light's view. As the light is far away, shadow aliasing appears in the view of the eye. (b) The shadow map is computed from the light's view after applying trapezoidal transformation to focus on the region (of only 15 plant models) which is potentially visible to the eye. As a result, high quality shadows are obtained.*

The second contribution of this paper is the recognition and a treatment of the *continuity problem* where the shadow map quality changes significantly from frame to frame resulting in the flickering of shadows. This occurs in all existing shadow map approaches that generate shadow maps of significantly different quality for small differences in the views of the eye or the light. Our approach computes a trapezoidal approximation so that there is a continuous change in the shape and size of trapezoid from frame to frame in order to control the transition in the shadow map resolution. See Figure 2 for an illustration.

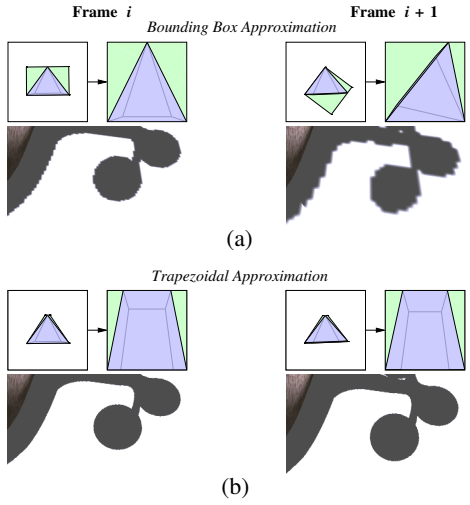

Figure 2: *(a) Flickering of shadows from one frame to the next generated by a bounding box approximation. (b) A smooth shadow transition generated with the use of our trapezoidal approximation. In each of the four pictures, the post-perspective space of the light is on the top left, the generated shadow map on the top right, and the shadow of a plant (as in the scene of Figure 1) on the bottom.*

The paper is organized as follows: Section 2 presents previous work; Section 3 discusses our reasons in choosing trapezoid to approximate the eye's frustum as seen from the light; Section 4 addresses the polygon offset problem; Section 5 describes the continuity problem; Section 6 details the computation of our trapezoidal approximation; Section 7 shows our results; and Section 8 concludes the paper with possible future works and limitations of our approach. Without loss of generality, our description assumes that there is a single light in the scene and the eye's frustum is completely within the light's frustum. It is straightforward to apply our approach to multiple lights (as used in Figure 7); Section 6.4 discusses the extension of our approach to the general case of eye and light.

2. Previous Work

In the following, we review works that are most relevant to ours in improving standard shadow maps.

The adaptive shadow map approach [FFBG01] addresses the resolution problem by using a hierarchical grid structure

instead of the standard "flat" shadow map. A great improvement to the shadow quality is gained, but it is currently not possible to map this approach to graphics hardware. Therefore, this approach is slow and not suitable for real-time applications.

The perspective shadow map approach (PSM) [SD02] tackles the problem of insufficient shadow map resolution in regions near to the eye by a non-uniform parameterization: A single (perspective) shadow map is generated to provide high resolution for objects nearby and low resolution for objects far away. To achieve this, the shadow map is calculated in the post-perspective space of the eye. This approach fundamentally improves shadow quality for some restricted cases (depending on the relative positions of the eye and the light) but meets a number of technical difficulties that hinder its applications to interactive and dynamic environments. First, the implementation of PSM is rather non-trivial with many tradeoffs to consider. For instance, it needs a robust implementation of a 3D convex hull algorithm together with union and intersection operations. Second, the polygon offset problem is worsened as depth values in the post-perspective space of the eye are distributed differently for various configurations between the eye and the light. As a result, a constant polygon offset may not be sufficient to avoid surface acne effects while maintaining good shadow quality.

The work of [BAS02] addresses the polygon offset problem with the linear distribution of depth values. In addition, it addresses the resolution problem by using the smallest box to bound objects in the eye's frustum as seen from the light. We argue in the next section that this parameterization is not necessarily ideal.

The recent work termed shadow silhouette maps [SCH03] attempts to combine the strengths of shadow volumes [Cro77] and shadow maps. It, however, also inherits the weaknesses of both techniques. Its reported frame rates does not seem to be good for interactive applications with moderate sizes of over tens of thousands triangles.

3. Increasing Shadow Map Resolution

A shadow map can be viewed simply to consist of two portions: one within and the other outside the eye's frustum. It is clear that only the former is useful in the determination of whether pixels are in shadow. Thus, to increase the shadow map resolution in one way is to minimize the entries, collectively termed as *wastage*, occupied by the latter. In other words, a good way to address the resolution problem is to better utilize the shadow map for the area within the eye's frustum as seen from the light, denoted as E; Figure 3, left. This requires the calculation of an additional normalization matrix N to transform the post-perspective space of the light to an N-space in general; see Figure 3, right, where N refers to the trapezoidal space and the bounding box space, respectively. Then, the shadow map is constructed from this N-space. With this, we need a minor modification to the standard shadow map algorithm: during shadow determination, a pixel is transformed into the N-space, rather than into the post-perspective space of the light, for the depth comparison. Except for the calculation of N which may be computationally expensive for some approaches, the shadow map generation and shadow determination still map well to graphics hardware.

Naïvely, the tighter the bound of the approximation to the area of interest, the better the resolution of the shadow map. The smallest such area is the 2D convex hull C of E. However, it is not clear how to transform efficiently C (which is a polygon of up to six edges) to some N-space while minimizing wastage. The next natural choice is to use the smallest bounding box B of C for the purpose (Figure 3, middle, bottom). A variant of this idea is presented by [BAS02] where C is the convex hull of those pixels in E occupied by objects in the scene. This approach trades the slow process of reading back pixels from the frame buffer for the high utilization of the shadow map memory. On the other hand, such bounding box approximation may not always result in minimizing wastage, as shown in Figure 3, right.

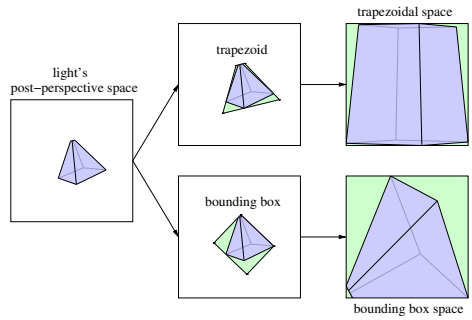

Figure 3: *An example of the trapezoidal approximation (middle, top) and the smallest bounding box approximation (middle, bottom) of the eye's frustum as seen from the light (left). The wastage in the shadow map generated by the former is much smaller than that by the latter in this case (right).*

In view of this, the next plausible choice is then to consider a general quadrilateral Q to approximate C. Low and Ilie [LI03] show a heuristic to compute Q deriving from the edges of the convex hull of C and having the smallest area bounding C. Such a Q is an approximation of the smallest area quadrilateral bounding C. The paper indicates but does not explicitly present and demonstrate a solution to the continuity problem. It does not seem straightforward to adapt the substantial calculation (from computer vision) of Low and Ilie to map specific regions in Q to a specific portion of the shadow map to control the continuous change in the shadow map resolution from frame to frame. In contrast, our proposed trapezoidal approximation of C discussed next uses

simple calculations, and it has a powerful control to address the continuity problem.

A trapezoid is recognized to be the most similar shape to E. Our work shows that the two parallel lines containing the top and the base edge of a trapezoid form a surprisingly powerful mechanism to control the shadow map resolution from frame to frame. This successfully addresses the continuity problem (see Section 6.1). Equally important and interesting for our choice of trapezoid are its two side edges in addressing another kind of "implicit" wastage not mentioned in the above discussion. Such wastage is the over-sampling of near objects in the shadow map where a lower sampling rate would suffice. We develop an efficient mechanism to decide on the two side edges to spread the available resolution to objects within a specified focus region (see Section 6.2). In comparison, the transformation used in the smallest bounding box does not have such flexibility in stretching a shape. As a result, the smallest bounding box has a deteriorating effect on the shadow map resolution when the depth of view increases.

The rest of this section formalizes the use of trapezoidal approximation in our approach. Consider a vertex v in the object space. Then, the vertex of v in the post-perspective space $\mathcal{L}$ of the light is $v_L = P_L \cdot C_L \cdot W \cdot v$ where P_L and C_L are the projection and camera matrices of the light and W is the world matrix of the vertex. The eight corner vertices of E are obtained from those corner vertices of the eye's frustum in the object space multiplied by $P_L \cdot C_L \cdot C_E^{-1}$ where C_E^{-1} is the inverse camera matrix of the eye. We treat E as a flattened 2D object on the front face of the light's unit cube. We use a trapezoid T to approximate (and contain) E; see Figure 3, middle, top. The normalization matrix N_T is constructed such that the four corners of T (as computed in Section 6) are mapped to a unit square. By applying N_T to $\mathcal{L}$, we transform the scene to the *trapezoidal space* $\mathcal{T}$. We call $v_T = N_T \cdot v_L$ a vertex in $\mathcal{T}$, N_T a *trapezoidal transformation*, and the shadow map derived from $\mathcal{T}$ a *trapezoidal shadow map*.

4. Handling Polygon Offset Problem

The intent of N_T is to transform only the x and y values of those vertices of objects within T. This transformation, however, also affects the z value of each vertex depending on its x and y values. As such, a good offset for each vertex depends on its x and y values, and thus the usual single offset for all vertices (as in the standard shadow map approach) is not adequate to remedy surface acne effects.

This problem can be solved with the idea of transforming only the x, y and w values of each vertex by N_T to $\mathcal{T}$, while maintaining the z value in $\mathcal{L}$. In this way, the polygon offset problem is not worsened and can be handled as in the case of a standard shadow map. As discussed in the next two paragraphs, this can be efficiently mapped to current graphics hardware supporting a programmable fragment stage.

In the first pass (shadow map generation), the vertex stage transforms each vertex v to $v_T = (x_T, y_T, z_T, w_T)$ and assigns $v_L = (x_L, y_L, z_L, w_L)$ as its texture coordinate. Note that texture coordinates over a triangle are obtained by linearly interpolating the v_L/w_T values of the vertices of the triangle. Next, the fragment stage replaces the depth of the fragment with z_L/w_L and adds to it an offset. In effect, we have set the z value of the vertex in $\mathcal{T}$ as z_L with the necessary polygon offset.

In the second pass (shadow determination), the vertex stage transforms each vertex to the post-perspective space of the eye as the output vertex. It also computes, for the vertex, two texture coordinates $v_L = (x_L, y_L, z_L, w_L)$ and $v_T = (x_T, y_T, z_T, w_T)$. Then, the fragment stage processes each fragment to determine shadow by comparing z_L/w_L to the value in the shadow map indexed by $(x_T/w_T, y_T/w_T)$.

We have two notes. First, a simpler and more efficient approach (to implement the idea of maintaining the z value in $\mathcal{L}$) to only keep v_T as $(x_T, y_T, \frac{z_L \cdot w_T}{w_L}, w_T)$ in the vertex stage (in both passes) does not always work. This problem is prominent in cases where, for example, the eye or light frusta contain large triangles. The reason is that such z is not correctly interpolated over each triangle as z is no longer expressible as some affine mapping of vertices in the world space. Second, the above vertex and fragment stages do slightly more work than that needed in the standard shadow map. Our experience with them for over 100K triangles remain highly interactive.

5. The Continuity Problem

As mentioned, the continuity problem is a consequence of a significant change in the shadow map quality from one frame to the next, resulting in flickering of shadows. For the smallest bounding box approach, the shadow map quality changes if there is a sudden change in the approximation of the eye's frustum as seen from the light. Figure 2(a) shows from frame i to frame $i+1$ that the orientation of the approximation of E with the smallest bounding box is changed. As a result, there is a drastic change to the resolution in different parts of the shadow map. In general, the problem can often occur when E transits from one shape to another different shape (where the number of side planes of the eye's frustum visible from the light's view is different). Additionally, the problem exists in [BAS02] as the smallest bounding box, enclosing all those pixels in E occupied by objects in the scene, changes drastically when some visible object is added or removed.

The continuity problem occurs in the perspective shadow map approach [SD02] as it relies on the convex hull of all objects that can cast shadows. This convex hull and the resulting shadow quality can change suddenly. In one case, this occurs when objects move into or out of the light's frustum in a dynamic environment. In another case, it can be observed when the algorithm virtually moves the position of the eye

to avoid, for example, the inverted order of objects due to the perspective projection. The continuity problem can occur in our trapezoidal approximation of E, too. However, we show in the next section that there exists an efficient and effective way to control the changes in trapezoids to address the problem.

6. Constructing Trapezoidal Approximation

Our aim is to construct a trapezoid to approximate E with the constraint that each such consecutive approximation results in a smooth transition of the shadow map resolution. Our strategy is to rely on a smooth transition in the shape and size of trapezoid to result in a smooth transition of the shadow map resolution. Section 6.1 discusses the computation to obtain the base and the top line. From these, the base and the top edge of the trapezoid are defined when the two side lines are computed as discussed in Section 6.2. Section 6.3 analyses the coverage of the focus region in the shadow map, and Section 6.4 extends our approach to handle scenarios where not all the eight vertices of the eye's frustum are inside the light's frustum.

6.1. Base and Top Lines

This step is to find two parallel lines in $\mathcal{L}$ to contain the base and the top edge of the required trapezoid. The aim is to choose the parallel lines such that there is a smooth transition when the eye moves (relative to the light) from frame to frame. We first present the algorithm and then discuss its rationale.

Step 1 Transform the eye's frustum into the post-perspective space $\mathcal{L}$ of the light to obtain E.

Step 2 Compute the center line l, which passes through the centers of the near and the far plane of E.

Step 3 Calculate the 2D convex hull of E (with at most six vertices on its boundary).

Step 4 Calculate the top line l_t that is orthogonal to l and touches the boundary of the convex hull of E. l_t intersects l at a point closer to the center of the near plane than that of the far plane of E.

Step 5 Calculate the base line l_b which is parallel to (and different from) the top line l_t (i.e., orthogonal to l too) and touches the boundary of the convex hull of E.

The above algorithm is such that the center line l governs the choices of l_t and l_b, with the exception for the case when the centers of the far and near planes (almost) are coincident. The algorithm handles that separately to result in the smallest box bounding the far plane as the desired trapezoid.

Imagine the eye's frustum is drawn within a sphere with the center of the sphere at the eye's position and the radius equal to the distance from the eye to each corner of the far plane. Suppose the eye's location does not change. Pitching and heading of the eye from one frame to the next can be encoded as a point (which is the intersection of l with the sphere) on the sphere to another nearby point, while rolling of the eye does not change the encoded point but results in a rotation of the eye's frustum along l. More importantly, with a smooth eye motion from frame to frame, the four corners of the far plane of the eye's frustum lying on the sphere also have a smooth transition on the sphere. As the positions of l and the mentioned four corners uniquely determine l_b, it also transits smoothly from frame to frame. Similarly, l_t transits smoothly from frame to frame, too.

Next, suppose the eye's location does change relative to the light from one frame to the next but maintains its orientation. In this case, it is only a matter of scaling E and the l_b and l_t computed are parallel to the previous ones. In other words, both l_b and l_t again transit smoothly from frame to frame under a smooth translation of the eye's frustum.

6.2. Side Lines

Before describing the computation of the side edges, we first analyze the effect of transforming a given trapezoid in Figure 4(a) by its N_T to $\mathcal{T}$. Note that N_T has the effect of stretching the top edge into a unit length. In this case, the top edge is relatively short compared to the base edge, and therefore the stretching results in pushing all the shown triangles towards the bottom of the unit square as in Figure 4(b). This means that the region near to the top edge (i.e., close to the near plane) eventually occupies a major part of the shadow map. This results in an over-sampling in the shadow map for objects very near to the eye while sacrificing resolution of the other objects (such as the second to the fourth triangle from the top). This is the kind of wastage due to over-sampling as mentioned in Section 3.

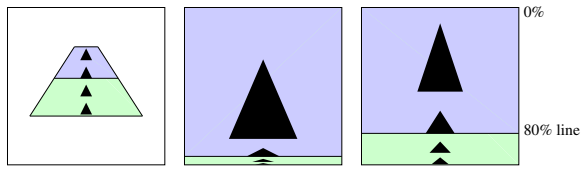

Figure 4: *For the trapezoid in (a), its corresponding $\mathcal{T}$ is shown in (b). In this case, we obtain an over-sampling for a small region of E. (c) For a different trapezoid computed with the 80% rule (having the same top and base lines), its trapezoidal transformation maps the focus region (the upper part of the trapezoid) to within the first 80% in the shadow map.*

Conversely, a small part of the shadow map is occupied by near objects when a "fat" trapezoid (having top and base edges of almost equal length) is transformed by its trapezoidal transformation. As our approach aims to achieve effective use of available shadow map memory by "important"

objects in the eye's frustum, we have designed our first version of the algorithm to compute the side lines and thus the required trapezoid as follows.

Assume the eye is more interested in objects and their shadows within the first δ distance from the near plane. That is, the region of focus, or simply the *focus region*, of the eye is the eye's frustum truncated at δ distance from the near plane. Let p be a point of δ distance away from the near plane with its corresponding point p_L lying on l in $\mathcal{L}$; see Figure 5. Let the distance of p_L from the top line be δ'. We design a trapezoid to contain E, so that N_T maps p_L to some point on the line of 80% or what we refer to as the *80% line* in $\mathcal{T}$ (see Figure 4(c)). Such an approach is termed the *80% rule*.

To do this, we formulate a perspective projection problem to compute the position of a point q on l with q as the center of projection to map p_L to a point on the 80% line $y = \xi$ (i.e. $\xi = -0.6$), and the base and the top line to $y = -1$ and $y = +1$, respectively. Let λ be the distance between the base and the top line. Then, the distance of q from the top line, denoted as η, is computed through the following 1D homogenous perspective projection:

$$\begin{pmatrix} \frac{-(\lambda+2\eta)}{\lambda} & \frac{2(\lambda+\eta)\eta}{\lambda} \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \delta'+\eta \\ 1 \end{pmatrix} = \begin{pmatrix} \tilde{\xi} \\ \omega \end{pmatrix},$$

$$\text{and} \quad \xi = \frac{\tilde{\xi}}{\omega}. \quad \text{So,} \quad \eta = \frac{\lambda\delta' + \lambda\delta'\xi}{\lambda - 2\delta' - \lambda\xi}.$$

Next, two lines passing through q and touching the convex hull of E are constructed to be the side lines containing the side edges of the required trapezoid.

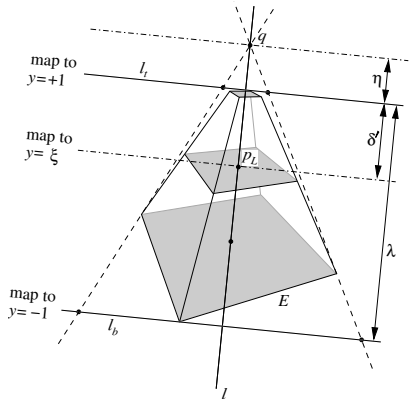

Figure 5: *A 1D homogenous perspective projection problem to compute q.*

For some situations (such as E of a dueling frusta case), the 80% rule can also result in a significant wastage of shadow map memory. We thus modify the above algorithm to an iterative process. Suppose the shadow map is a map with x horizontal lines of entries. (In our experiments, x is 1024 or 2048.) In the first iteration, p_L is mapped to the 80% line (or $0.8x$), and in each subsequent iteration, p_L is mapped to an entry one line before that of the last iteration to compute q. With each computed q, we have a corresponding trapezoid and its trapezoidal transformation N_T computed as before. From all the iterations, we adopt the trapezoid with its N_T that transforms the focus region to cover the largest area (though other metrics are possible) in the shadow map. Note that the above computation is not expensive as it involves simple arithmetic and only a small number of iterations. In fact, for a given up vector of the eye and a given angle between the eye's and the light's line of sight, the best ξ to where p_L is mapped is independent of the scene and can thus be pre-computed. Therefore, all these best ξ (and thus η) can be stored in a table with the parameter of the angle between the eye's and the light's line of sight, for each possible up vector of the eye. Thus, a simple table lookup can also replace the above iterative process.

6.3. Focus Region in the Shadow Maps

To understand our 80% rule, we generate a plot (as shown in Figure 6) of the total area covered by the focus region in the shadow map by varying the angle (represented as a data point on the xy-plane) between the eye's and the light's line of sight while keeping the up vector constant. We have also experimented with a series of the same kind of plots with different up vectors. We observe that consecutive plots of slightly different up vectors are surfaces of very close values. These plots indicate that there is a smooth transition on the area occupied by the focus region (though one could possibly attempt to prove it formally). This is a strong indication that our approach addresses the continuity problem well. From our experience, we noted that the 80% rule is effective. Nevertheless, one can adjust this percentage according to the need of the application.

6.4. General Case of Eye and Light

The above discussion assumes that the eye's frustum lies completely within the light's frustum, such as in an outdoor scene where the sun is the main light source. If this is not the case, one naïve adaptation is to enlarge the light's view to include the eye's frustum. This is not an effective use of the shadow map. Also, this can be delicate to handle and may not always be feasible: there are situations where the vertices of the eye's frustum lie behind or on the plane passing through the center of projection of the light and parallel to the near plane of the light. Such vertices have inverted order or are mapped to infinity in $\mathcal{L}$. Instead, the next two paragraphs discuss a simple extension without such undesirable situations.

Specifically, it suffices to only transform the portion of

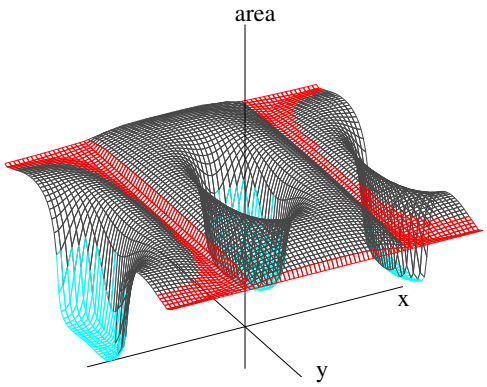

Figure 6: *A plot of the areas occupied by the focus regions in the shadow map with a constant up vector of the eye while varying the angle between the eye's and the light's line of sight. The focus regions occupy small areas for the dueling frusta case, but large areas when, for example, one side face of E is visible in the light's view.*

the eye's frustum that is inside the light's frustum to $\mathcal{L}$. The remaining portion which is not inside the light's frustum is clearly not illuminated and hence cannot have shadows. Therefore, our approach only needs to process the intersection I between the light's frustum and the eye's frustum (with no more than 16 intersections as its vertices). This conveniently avoids the above problem due to the perspective transformation. Our algorithm is also adapted accordingly to take care of the following issues while maintaining a good control of the continuity problem.

First, the line l passing through the centers of the near and the far plane of the eye's frustum may no longer be the center line for the computation of the base and the top line. One approach is to compute the center point e of the vertices of I, and use the line passing through the position of the eye and e to be the new center line l for the computation. Second, a new focus region has to be defined, because the focus region may not be completely within I. One approach is to geometrically push the near and the far plane of the eye (closer to each other) to tightly bound I in the world space to obtain f' as the distance between those planes. Let f be the distance between the original far and near planes of the eye in the world space. Then, the new focus region lies within the new near plane and its parallel plane, where the distance between the planes is $(\delta \cdot f'/f)$. Note that δ is the distance originally chosen to set the focus region (as defined in Section 6.2).

With the above, our approach is now suited for a wider range of applications: near to far lights, and both indoor and outdoor scenes. The accompanying video (available at our project webpage: www.comp.nus.edu.sg/∼tants/tsm.html) shows an animation of such cases with two lights illuminating a fantasy character (Figure 7). The video shows that our approach can achieve high shadow quality for the close

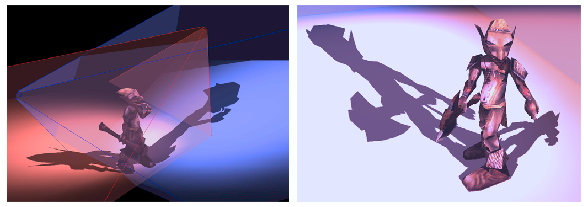

Figure 7: *On the left, the character is lit by two nearby lights as viewed from outside the lights' frusta. On the right, the character is lit by a nearby light (left shadow) and a far light (right shadow).*

light situation suitable for the standard shadow map as well as for the transition to the far light situation unfavorable to the standard shadow map.

7. Implementation and Results

We have implemented the proposed trapezoidal shadow maps using GNU C++ and OpenGL under Linux environment on an Intel Pentium 4 1.8GHz CPU with a nVidia GeForce FX5900 ultra graphics controller. We use ARB vertex/fragment programs to address the polygon offset problem. The shadow maps are rendered into a pbuffer using GLX_SGIX_pbuffer. Note that our approach uses various geometric yet simple operations such as convex hulls, line operations etc. in 2D. Robustness issues are easy to handle in our 2D cases. The standard shadow maps (SSM) [Wil78], a version of the smallest bounding box approximation (BB) (see [BAS02]), and the perspective shadow maps (PSM) [SD02] are implemented for purposes of comparison. Details of our PSM implementation are provided in our project webpage (see [Koz04] for possible improvements to PSM, and also [WSP04]).

7.1. Fantasy World

Our first experiment is on a fantasy world with over 100K triangles, and uses a shadow map size of 1024x1024. All objects can cast shadows where the only light source is set at a far distance from the scene. Figure 8 (as shown in the color plates) shows snapshots of our scene rendered by the various approaches. The objects in the scene include a static tree, three mushrooms, a pergola, dynamic objects inclusive of a dragon and a few characters, and a lotus.

The accompanying video contains flythroughs of the scene by the various approaches. All the approaches run smoothly with an average of 28 frames per second without any special culling and optimization. With SSM, shadows of static objects do not flicker while shadows of dynamic objects often flicker. With BB, the flickering occurs for both static and dynamic objects. The quality of the shadows is not good in general, and the flythrough experience is not acceptable. With PSM, it produces better shadow quality than that

of BB for its favorable cases. It, however, converges to the SSM for those unfavorable cases such as when the position of the eye needs to be moved back so that all objects casting shadows into the eye's frustum are included in the new "virtual" eye's frustum. With TSM, the animation is smooth with only occasional flickering of shadows. The experience of the flythrough is very pleasant.

7.2. Urban Model

Our second experiment is on an urban model with approximately 1.4 million triangles from 79 objects of buildings, vehicles etc. where the only light source is set at a far distance from the scene. We use a shadow map size of 2048x2048 pixels with the eye having a large depth of view to test the quality of shadows of objects from near to far. Such a setting is unfavorable to BB (as mentioned in Section 3), and comparison with BB is thus not necessary and not shown here.

The accompanying video shows flythroughs of the scene and comparisons between PSM and TSM. See also Figure 9 (as shown in the color plates) for comparisons of shadows between PSM and TSM. We observe that both far away and nearby shadows generated by TSM are of better quality than those by PSM. Our program for this scene uses ARB_occlusion_query for a simple occlusion culling in the shadow map generation step. On the average, TSM renders only about 56 objects during the shadow map generation. In contrast, SSM has to render all of the 79 objects, and BB and PSM about 61 and 72 objects on the average, respectively. The frame rates for all approaches are small due to the large number of triangles in the scene while there is no sophisticated optimization in our implementation.

8. Concluding Remarks and Limitations

We propose the novel trapezoidal shadow maps for real-time interactive applications. Our implementation shows that it is practical and maps well to graphics hardware. We note that our approach is only one way to address the resolution and the continuity problem. It is a reasonable heuristic to generate shadow maps of good resolution, but the issues on over- and under-sampling remain for various situations such as in the dueling frusta case where the trapezoidal approximation does not have any particular advantage over other approximations. The possibility to compute an optimal resolution by using only one shadow map remains an open question. Also, the approach addresses the continuity problem due to the motions of the eye and the light but not that of objects per se. Nevertheless, we take comfort in a good overall control of the shadow map resolution and its smooth transition, so that shadows blend well into the scenes without attracting special (undesirable) attention!

Acknowledgements

We would like to thank John Cannon for helpful discussions. The character models used in our experiments are obtained from www.planetquake.com. This research is supported by NUS under grant R-252-000-181-112.

References

[BAS02] BRABEC S., ANNEN T., SEIDEL H.-P.: Practical shadow mapping. *Journal of Graphics Tools 7*, 4 (2002), pp. 9–18.

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. In *Proceedings of SIGGRAPH* (1977), pp. 242–248.

[FFBG01] FERNANDO R., FERNANDEZ S., BALA K., GREENBERG D. P.: Adaptive shadow maps. In *Proceedings of SIGGRAPH* (2001), pp. 387–390.

[Koz04] KOZLOV S.: Perspective shadow maps: Care and feeding. *In GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics* (2004), pp. 217–244.

[LI03] LOW K.-L., ILIE A.: Computing a view frustum to maximize an object's image area. *Journal of Graphics Tools 8*, 1 (2003), pp. 3–15.

[SCH03] SEN P., CAMMARANO M., HANRAHAN P.: Shadow silhouette maps. In *Proceedings of SIGGRAPH* (2003), pp. 521–526.

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Proceedings of SIGGRAPH* (2002), pp. 557–562.

[SKvW*92] SEGAL M., KOROBKIN C., VAN WIDENFELT R., FORAN J., HAEBERLI P.: Fast shadows and lighting effects using texture mapping. In *Proceedings of SIGGRAPH* (1992), pp. 249–252.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH* (1978), pp. 270–274.

[WM94] WANG Y., MOLNAR S.: *Second-Depth Shadow Mapping*. Tech. Rep. TR94-019, Department of Computer Science, University of North Carolina at Chapel Hill, 1994.

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

[WSP04] WIMMER M., SCHERZER D., PURGATHOFER W.: Light space perspective shadow maps. In *Proceedings of Eurographics Symposium on Rendering* (2004).