Parallel-Split Shadow Maps for Large-scale Virtual Environments

Fan Zhang Hanqiu Sun Leilei Xu Lee Kit Lun [*]
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Abstract

Shadowing effects dramatically enhance the realism of virtual environments by providing useful visual cues. Shadow mapping is an efficient algorithm for real-time shadow rendering, which is extensively adopted in real-time applications by its generality and efficiency. However, shadow mapping usually suffers from the inherent aliasing errors due to the image-based nature. In this paper, we present the Parallel-Split Shadow Maps (PSSMs) scheme, which splits the view frustum into different parts by using the planes parallel to the view plane and then generates multiple smaller shadow maps for the split parts. A fast and robust split strategy based on the analysis of shadow map aliasing has been proposed, which produces a moderate aliasing distribution over the whole depth range. By applying the geometry approximation procedure to each of the split parts instead of the whole scene, the tighter bounding shapes of visible objects enhance the utilization of the shadow map resolution. Hardware-acceleration is used to remove the extra rendering passes when synthesizing the scene-shadows. Our approach is intuitive to implement without using complex data structures, with real-time performance for dynamic and large-scale virtual environments.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—[Anti-aliasing, bitmap and frame-buffer operations] I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—[Color, shading, shadowing, and texture]

Keywords: Shadow Mapping, Image-based Rendering, Large-scale Virtual Environments

1 Introduction

Shadows are essential for the realism of computer images and virtual environments. Even tremendous advances in computer graphics and programmable GPUs have made real-time photo-realistic rendering a reality, the synthesis of realistic shadowing effects is still challenging and computationally intensive.

The two main categories of shadow algorithms are *shadow volumes* [Crow 1977] and *shadow mapping* [Williams 1978]. Shadow volumes work in the object space, in which the connectivity information of all polygons is required to compute the silhouette of each shadow caster. The scene complexity thus directly influence the performance due to the object-based nature. Shadow mapping, on the other hand, works in the image space and requires only one additional rendering pass per light. With such flexibility and performance, shadow mapping is extensively adopted in real-time applications. Shadow mapping is basically a two-pass algorithm. In the first pass, the scene is rendered from the light's point of view. The scene's distance information to light is stored in the current depth buffer (i.e. shadow map). In the second pass, the scene is rendered from the eye's point of view. Here, each pixel is transformed into the light's view space again. If the depth value of the pixel is larger than the depth value stored in the shadow map, this pixel is shadowed.

As any image-space method based on discrete buffers, shadow mapping usually suffers from aliasing problems. The aliasing errors are mainly caused by the insufficient shadow map resolution (*undersampling*). Such aliasing errors can be further classified into *perspective aliasing* and *projection aliasing* [Stamminger and Drettakis 2002] and an insightful analysis for different perspective reparameterizations is presented in [Wimmer et al. 2004]. In the following, the shortcomings of shadow mapping methods when applied especially for large-scale virtual environments are discussed:

Texture Resolution: In large-scale virtual environments, a very-large shadow map resolution is required to produce high-quality shadowing effects. However, it's impractical and insufficient to use a single huge shadow map because of the limitation of the maximum texture resolution on commodity hardware (e.g. 2048x2048 at maximum on today's DirectX9-level hardware).

Global Reparameterizations: Global reparameterizations treat all objects in the same manner, such as warping the shadow map plane by using the perspective transform. Due to the complexity of the scene, it is however impractical to produce sufficient sampling densities for pixels being mapped by reparameterizing a single shadow map, especially in large-scale virtual environments.

Geometry Approximation: The *geometry approximation* procedure is usually applied to increase the available shadow map resolution, which makes the shadow map focusing on the objects within the view frustum as seen from the light as possible. In large-scale virtual environments, the empty gaps between objects in the *global* approximation of the *whole* scene may waste a great amount of resolution.

Dueling Frusta Case: Few shadow mapping approaches handle the worst case *dueling frusta*, in which the directions of viewing and lighting are nearly opposite. In this extreme case, the visible region of view frustum only occupies a small portion of the shadow map. The anti-aliasing method based on reparameterizing a single shadow map still remains an open question.

In this paper, we develop the *parallel-split shadow maps* (PSSMs) scheme, which splits the view frustum into different depth layers by using split planes parallel to the view plane, and then render an independent shadow map for each of the split parts (Figure 1). Our approach is based on the observation that points at different distance to the viewpoint need different shadow map sampling densities. This view-driven characteristic of the shadow map resolution motivates our approach to avoid the above shortcomings due to the facts: 1) Each of the split parts has an independent shadow map, different parameterizations can be applied in different depth ranges according to the application's requirement. The single shadow map texture

[*]{e-mail: fzhang, hanqiu, llxu, kllee2}@cse.cuhk.edu.hk

with a large size is replaced by multiple textures with smaller sizes, and the texture memory used in our PSSMs scheme is usually less than the single shadow map used by other shadow mapping algorithms but with high-quality shadow rendering. 2) Since the depth range is split into smaller layers, our split scheme changes the resolution requirement from *"sufficient for every point"* to *"sufficient for every depth layer"*. 3) The geometry approximation is applied in each of the depth ranges *separately*. The tighter bounding shape significantly enhances the utilization of the shadow map resolution. 4) Because each shadow map in PSSMs is focused in smaller sub-frusta, the shadow qualities in the dueling frusta case is also greatly improved.

In the following, Section 2 outlines the previous related work in shadow rendering. Section 3 describes our PSSMs scheme for large-scale virtual environments, including pre-processing of lighting sources, view frustum split, light's frustum split, PSSMs rendering and scene-shadows rendering. Section 4 presents the experimental results of our proposed scheme, in comparison with single-map shadow mapping methods for several dynamic virtual environments. Finally, the summary and further work go to Section 5.

2 Previous work

The literature on real-time shadow rendering in computer graphics is vast, and far beyond the scope of this paper. We thus concentrate on most relevant to our algorithm in solving the aliasing problem of shadow mapping. More details related to real-time shadow rendering in general are given in the comprehensive survey [Woo et al. 1990].

Adaptive Shadow Maps (ASMs) [Fernando et al. 2001] reduces aliasing by storing the shadow map as a hierarchical grid structure. By evaluating the contributions of shadow map pixels to the overall image quality, it is refined to create higher resolution at regions that contain shadow boundaries. Artifacts at those critical regions is thus greatly reduced without requiring a "flat" shadow map of huge resolution. However, the traversal and refinement operations require many rendering passes and aren't feasible for real-time applications. Recently, a GPU-accelerated implementation of ASMs was presented by Lefohn et al. [Lefohn et al. 2005], but the reported performance is still not satisfactory in complex virtual environments. The work presented by Arvo [Arvo 2004] can be regarded as a simplified variant of ASMs, which proposed to use a tiled grid data structure to tessellate the light's viewport. For each cell area in this grid, a particular sampling density is adopted based on a heuristical analysis. However, subdivision artifacts may occur due to the heuristic nature of the binary cutting, such that some tiles may obtain extremely non-square shapes from the shadow map.

Perspective reparameterizations have been first proposed in Perspective Shadow Maps (PSMs) [Stamminger and Drettakis 2002], in which both the scene and the light source are transformed to the post-perspective space. Then the shadow map is generated in this space by rendering a view from the transformed light source to the unit cube (i.e. transformed view frustum). Since the shadow map "sees" the scene after perspective projection, perspective aliasing can be significantly decreased. [Kozlov 2004] presented a practical implementation of PSMs. A recently improved perspective reparameterization is the Light Space Perspective Shadow Maps (LiSPSMs) [Wimmer et al. 2004], which avoids some inconvenience of PSMs and leverages the aliasing distribution over the whole depth range. Trapezoidal Shadow Maps (TSMs) [Martin and Tan 2004] is another perspective reparameterization and very similar to PSMs and LiSPSMs. The essential difference between TSM and prior perspective parameterizations is that a different perspective warping transform is used in TSM, such that the user-focused portion at the front of the frustum is mapped to the 80% line on the shadow plane. Another way of warping the shadow map texels is to tilt the shadow plane [Chong and Gortler 2004][Low and Ilie 2003], but such oblique shadow plane can not be globally optimal to the whole scene.

Tadamura et al. [Tadamura et al. 2001] proposed a "plural" approach which uses a dynamic texture array comprised of multiple shadow maps with varying resolutions. Like the approach proposed in this paper, it divides the view frustum into several parts to approximate the continuously varying of the resolution along the distance from the view point. Comparing to this approach, three important improvements are proposed in this paper: (1) The complicated and time-consuming computations for optimal lengths of split parts are replaced by our efficient split scheme. A straight and robust split scheme based on the analysis of perspective parameterizations has been proposed in this paper. Our split scheme satisfies the depth adaptive requirement of the shadow map resolution, but without the recursive searching procedure. The shadow quality only depends on the interactively adjustable number of split layers (given the shadow map resolution). (2) The varying resolutions of shadow maps in the Tadamura approach is not compatible with the hardware that only support the textures with the dimension of "power of two". Instead, the same resolution configuration makes our approach more general and easy to control the overall texture memory required. (3) The Tadamura et al. approach requires multiple rendering passes for both shadow maps rendering and scene-shadows rendering. In our approach, only single pass is needed for scene-shadows rendering by using pixel shader on programmable GPUs. The potential acceleration of shadow maps rendering on future hardware is also discussed.

Practical Shadow Mapping [Brabec et al. 2002] proposed to calculate the bounding box (geometry approximation) of the view frustum. The tight fitting frustum makes the shadow map only concentrates on the visible parts of the scene, the available shadow map resolution is thus increased. Unlike other shadow-mapping methods which store a single depth at each pixel, Deep Shadow Maps (DSMs) [Lokovic and Veach 2000] stores a representation of the fractional visibility through a pixel at all possible depths. Second Depth Shadow Mapping [Wang and Molnar 1994] can be used to reduce problems due to depth quantization and self occlusions. Percentage Closer Filtering [Reeves et al. 1987] extends the the concept of classical bilinear filtering used in texture sampling, to help anti-aliasing around shadow boundaries. Recent techniques [Sen et al. 2003][Govindaraju et al. 2003] propose to integrate shadow maps and shadow volumes or other primitives to inherit the merits of both image-space and object-space methods.

3 Parallel-Split Shadow Maps

The goal of shadow map parameterizations is to produce an optimized distribution of shadow map texels, in order to providing sufficient sampling densities for points being mapped. Due to the complexity of the scene, it is however impractical to achieve this goal by reparameterizing a single shadow map especially in large-scale virtual environments. Thus, our PSSMs scheme treats the continuous depth range as multiple discrete layers. The geometry approximation procedure and different parameterizations can be applied to each layer separately. Our approach can produce a more even aliasing distribution and tighter bounding shapes such that utilizing the shadow map resolution better. For the clarity of illustration, we split the view frustum into 3 parts in Figure 1. The general case of

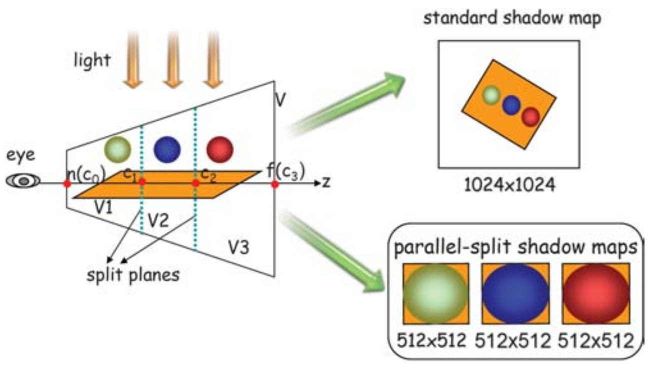


Figure 1: Split the view frustum into three parts, and shadow maps with the same resolution are generated for the split parts.

splitting is discussed in subsection 3.3.

3.1 Light Sources

In our system, we use directional lighting sources to illustrate the idea for clarity. Actually, all kinds of lighting sources including point lighting sources can be unified as directional lighting sources in the light's post-perspective space (Figure 2). After applying the light's projection transform to the scene, all objects are transformed to the normalized clip space of the lighting source. Point lighting sources are converted to directional ones and directional lighting sources keep unchanged. The shadow maps are then rendered in this space, and all pixels also need to be transformed into this space during shadow determination. Therefore, by rendering the shadow maps and scene-shadows in the light's post-perspective space, our system supports both the directional and point lighting sources. The pre-processing on light sources simplifies our discussion later, which was also used in LiSPSMs.

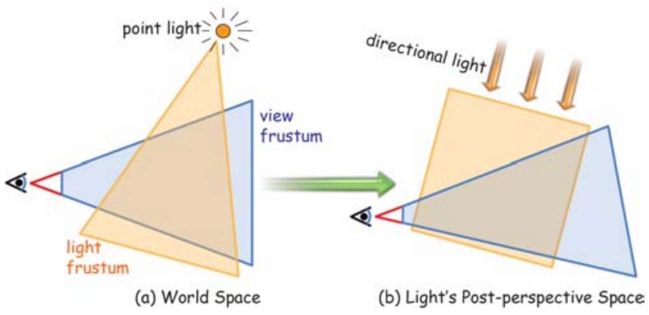


Figure 2: Point lighting sources are converted to directional ones in light's post-perspective space.

3.2 PSSMs Overview

The processing steps of the PSSMs scheme are outlined in the following:

STEP 1 Split the view frustum into multiple depth parts.

STEP 2 Split the light's frustum into multiple smaller ones, each of which covers one split part also the objects potentially casting shadows into the part.

STEP 3 Render a shadow map for each split part.

STEP 4 Render scene shadows for the whole scene.

Here frequently used notations in our discussion are listed:

notation	**description**
V	view frustum.
V_i	the ith split part of $V (1 \leq i \leq m)$.
n and f	near and far planes of V.
m	number of split parts.
C_i	depth of the ith split plane in the view space $(1 \leq i \leq m-1)$, for convenience we supplement to define $C_0 = n$ and $C_m = f$.
T_i	the shadow map texture for V_i.
PSSM$(m; [res])$	the split scheme to split V into m parts, and the resolution of each shadow map is res.

3.3 View Frustum Split

The "split" idea is based on the observation that objects located in different depth layers to the viewer need different texture resolutions. Figure 3 shows the m split planes in general that are located at $\{C_i \mid 0 \leq i \leq m\}$ along z axis.

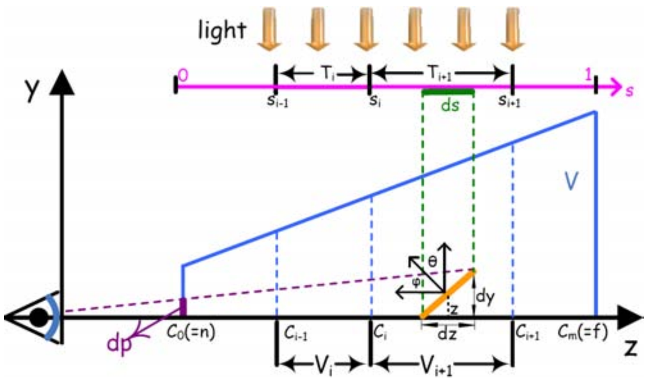


Figure 3: Along the z axis the view frustum is split into parts by using the split planes at $\{C_i \mid 0 \leq i \leq m\}$.

The key problem here is how to specify the split positions. Before discussing how to select C_i, we briefly outline the main causes for shadow map aliasing errors which are also presented in [Wimmer et al. 2004]. In Figure 3, the light beams through a texel with the size $\mathrm{d}s \times \mathrm{d}s$ in *normalized* texture space (i.e. $s \in [0,1]$) falls on a surface with the length $\mathrm{d}z$ in world space (e.g. $\mathrm{d}z = (f-n)\mathrm{d}s$ for standard shadow maps). The size of view beams $\mathrm{d}p$ on the screen projected from the surface is approximately $n\mathrm{d}y/z$. φ and θ denote the angles between the surface normal and vector to the screen and shadow map plane respectively. Since $\mathrm{d}y = \mathrm{d}z\cos\varphi/\cos\theta$, the shadow map aliasing error $\mathrm{d}p/\mathrm{d}s$ can be expressed as

$$\frac{\mathrm{d}p}{\mathrm{d}s} = n\frac{\mathrm{d}z}{z\mathrm{d}s}\frac{\cos\varphi}{\cos\theta} \qquad (1)$$

Shadow map *undersampling* appears when $\mathrm{d}p$ is larger than the pixel size of the screen, this can happen when *perspective aliasing* $\mathrm{d}z/z\mathrm{d}s$ or *projection aliasing* $\cos\varphi/\cos\theta$ becomes large. Projection aliasing $\cos\varphi/\cos\theta$ usually happens for a surface almost parallel to the light direction. Since projection aliasing is heavily related to the scene's geometry details, the local increase of sampling densities on this surface is required to reduce this category of aliasing.

This is not possible without an inevitably expensive scene analysis at each frame and complex data structure, which can not be accelerated by current hardware. On the other hand, perspective aliasing $\mathrm{d}z/z ds$ comes from the perspective foreshortening effect. Perspective aliasing can be reduced by applying a global transformation to warp the shadow map texels. In this paper, our split scheme is based on the analysis of the perspective aliasing distribution.

Now we present our selection criteria for $\{C_i\}$ in detail. In the following, three split schemes are discussed: **logarithmic split scheme**, **uniform split scheme** and **practical split scheme**. *Logarithmic split scheme* is designed to provide the theoretically even distribution of perspective aliasing errors. *Uniform split scheme* produces the worst distribution of perspective aliasing errors as the uniform sampling densities same to standard shadow maps. In the practical applications, we integrate *logarithmic split scheme* and *uniform split scheme* into *practical split scheme* to produce a moderate sampling densities over the whole depth range.

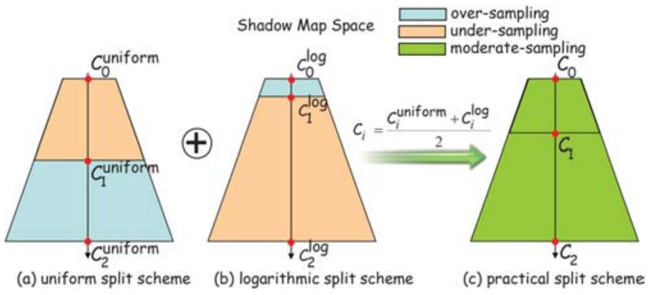


Figure 4: Practical Split Scheme.

Logarithmic Split Scheme $\{C_i^{log}\}$:

Ideally, the optimal distribution of perspective errors makes $\mathrm{d}p/\mathrm{d}s$ constant over the whole depth range. That is namely $\mathrm{d}z/z\mathrm{d}s = \rho$ (ignoring projection aliasing), where ρ is a constant. Then,

$$s = \int_0^s \mathrm{d}s = \frac{1}{\rho}\int_n^z \frac{\mathrm{d}z}{z} = \frac{1}{\rho}\ln(z/n)$$

Here we have an important notice: the local parameterization of the shadow map $s \in [0,1]$ implies an assumption – the shadow map accurately covers the view frustum and no any resolution is wasted on invisible parts of the scene. That means the above parameterization only can be achieved in theoretical. Based on the assumption of $s \in [0,1]$, easily we have $\rho = \ln(f/n)$. Hence, the theoretically optimal perspective shadow map parameterization $s(z)$ should satisfy the following logarithmic function:

$$s = \frac{\ln(z/n)}{\ln(f/n)} \tag{2}$$

However, the only non-linear transform supported by current hardware is the perspective transform ($s = A/z + B$). To approximate the logarithmic function, we have to discretize it at several depth layers. More layers we split the view frustum into, more optimal approximation of this function we have. This is the essential motivation of the "split" idea we proposed in this paper.

Equation (2) can be discretized at $z = \{C_i^{log}\}$:

$$s_i = s(C_i^{log}) = \frac{\ln(C_i^{log}/n)}{\ln(f/n)}$$

or

$$C_i^{log} = n(f/n)^{s_i} \tag{3}$$

Because this split scheme is designed to produce the theoretically even distribution of perspective aliasing errors, the resolution allocated for $[C_i, C_{i-1}]$ should be i/m of the overall texture resolution. Substituting $s_i = i/m$ in Equation (3) gives:

$$C_i^{log} = n(f/n)^{i/m} \tag{4}$$

The main drawback of this split scheme is that the lengths of split parts near the viewer are too small, so few objects can be included in these split parts. This is due to the theoretically optimal parameterization assumes that the shadow map accurately covers the view frustum and no any resolution is wasted on invisible parts of the scene. In other words, this assumption requires every $z \in [n, f]$ must be mapped to a unique $s \in [0,1]$ in the normalized texture space. However, this assumption can't be satisfied in practice such that over-sampling occurs for the parts nearer the viewer, and under-sampling for the parts further from the viewer. As a result, logarithmic split scheme is hard to be applied in practice.

Uniform Split Scheme $\{C_i^{uniform}\}$:

The simplest split scheme is to place the split planes uniformly along the z axis:

$$C_i^{uniform} = n + (f-n)i/m. \tag{5}$$

The produced distribution of perspective aliasing errors is the same as for standard shadow maps. Because $s = (z-n)/(f-n)$ in standard shadow maps, the perspective aliasing is ($\doteq$ means ignoring projection aliasing):

$$\frac{\mathrm{d}p}{\mathrm{d}s} \doteq n\frac{\mathrm{d}z}{z\mathrm{d}s} = \frac{n(f-n)}{z}$$

That means the perspective aliasing in uniform reparameterizations increases hyperbolically as the object moves near to the view plane. Therefore, uniform split scheme results in under-sampling at the points near the viewer, over-sampling at the points further from the viewer. On the contrary to logarithmic split scheme provides the *theoretically optimal* aliasing distribution, uniform split scheme results in the *theoretically worst* aliasing distribution.

Practical Split Scheme $\{C_i\}$:

Since logarithmic and uniform split schemes can't produce appropriate sampling densities for both near and far depth layers simultaneously, practical split scheme is designed to moderate the *theoretically optimal* and *worst* sampling densities in the above two *extreme* split schemes. It simply adopts the even value of C_i^{log} and $C_i^{uniform}$ as the split positions C_i to achieve a better aliasing distribution.

$$C_i = (C_i^{log} + C_i^{uniform})/2 \tag{6}$$

In Figure 4, practical split scheme produces moderate sampling densities for both near and far split parts. Combining Equations (4), (5) and (6), the split positions C_i can be calculated by the following equation,

$$C_i = \frac{n(f/n)^{i/m} + n + (f-n)i/m}{2} + \delta_{bias}, \quad \forall 0 \le i \le m. \tag{7}$$

Where the variable δ_{bias} is a non-negative bias, which can be used to accurately adjust the clip positions according to practical requirements of applications. In [Tadamura et al. 2001], an adjustable parameter has been introduced to determine the appropriate shadow map resolution to alleviate artifacts, through an expensive scene analysis at each frame. However, the tradeoff between shadow qualities and rendering speed is prone to be broken, due to the fact that the calculation time for shadows is proportional to the value of this parameter. Instead, Equation (7) gives a fast and robust split scheme without involved computations for both resolutions and split positions.

3.4 Light's Frustum Split

In this step, we split the light's frustum W into smaller ones $\{W_i\}$. Each W_i can be constructed by computing the axis-aligned bounding box (in the light's view space) of $V_i \cap W$. W_i covers V_i and the objects potentially casting shadows into V_i.

To simplify our explanations, the direction of the lighting source is perpendicular with the view direction in Figure 5. B_i and B denote the bounding box of V_i and V respectively. To utilize the shadow map resolution better, W_i and W can be focused to B_i and B respectively. We call B_i (or B) as the *geometry approximation* of V_i (or V), which makes the shadow map only focusing on "shadow relevant" objects.

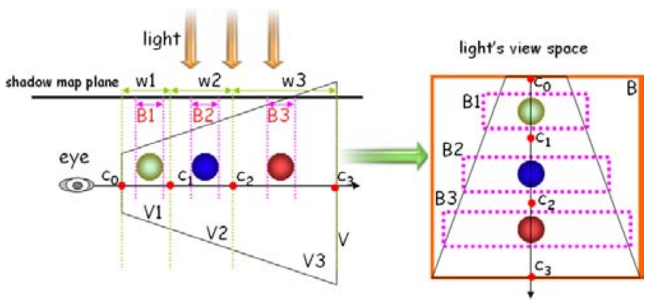


Figure 5: The light's frustum W is split into $\{W_i\}$, each of them covers the split part V_i. Then, applying geometry approximation to W_i to produce a focused bounding shape B_i. The sum of B_i is smaller than the bounding shape B of V.

The global geometry approximation methods focus the whole scene V to B. A main shortfall of the global approximation methods is that they don't take into account the distribution of objects, the resulted bounding shape makes the shadow map waste the resolution on the empty gaps between objects. This situation becomes worse when the objects are located sparsely in large-scale scenes. Stefan Brabec et al. [Brabec et al. 2002] proposed to utilize the histogram equalization to uniform the distribution of the objects, but it needs to read back the frame buffer again such that the performance drops down.

On the contrary, our PSSMs scheme applies the geometry approximation approach to each of the split parts separately, rather than the whole view frustum. In Figure 5, *axis-aligned bounding box* (AABB) is used as an example of geometry approximation methods. Apparently, $\sum_{i=1}^{m} Area(B_i) \leq Area(B)$. This means the more precise approximations can be achieved based on our split scheme. More parts we split the view frustum into, more compact bounding shapes we have. This *partial* (or *non-global*) approximation scheme can be easily integrated with other shadow mapping techniques.

3.5 PSSMs Rendering

After the light's frustum is split into $\{W_i\}$, each split part V_i is rendered to a shadow map T_i in the space W_i. We call $\{T_i\}$ as parallel-split shadow maps (PSSMs). Since *practical split scheme* (Equation 6) only moderates the split positions of *logarithmic split scheme* (Equation 5) and *uniform split scheme* (Equation 4), the resolutions of $\{T_i\}$ are still the same in our implementations. In comparison with the shadow map resolution in single-map methods is usually above 1024x1024, PSSM(m; 512x512) (m is less than 4 in practice) satisfies the requirements of most applications. The total texture memory used (512x512xm) in PSSMs is less than that (1024x1024x1) in single-map methods. Furthermore, other shadow mapping techniques can be seamlessly integrated into our split scheme.

Since our approach utilizes multiple shadow maps to improve scene-shadows qualities, the generation of PSSMs needs multiple passes which may result in the performance degrading. For this concern, *multiple rendering targets* (MRT) in the specification of OpenGL 2.0 (or Microsoft Direct3D 9.0) might be the choice for this problem, which can render split parts into different textures simultaneously. In many applications especially for the large-scale virtual environments, it is necessary to support the high quality scene-shadows rendering with the optimal texture memory cost and real-time performance by using the up-to-date graphics hardware. Even without MRT support, it is worth to pay a moderate cost on the multi-pass rendering in real-time with the dramatic improvement of shadow qualities. In our experiments, we have tested the PSSM(2), PSSM(3) and PSSM(4) schemes which produce a good tradeoff between the rendering speed and scene-shadows qualities. Particularly, PSSM(3) produces the best result in terms of both performance and shadow qualities. Our experimental results ($m \leq 4$) have shown the better visual qualities with less texture memory used and real-time performance. Table 1 shows the rendering speed of the above PSSMs schemes in our experiments.

3.6 Scene-shadows Rendering

With the PSSMs generated in the previous step, shadow effects can be now synthesized into the virtual scene. Like standard shadow mapping, each pixel should be transformed into the light space when determining if the pixel is shadowed or not. The differences here are: 1) we should select the correct shadow map T_i, and 2) the pixel should be transformed into W_i rather W. This procedure is accelerated by hardware.

Theoretically, multiple shadow maps require extra rendering passes in this step because we need to sample all split shadow maps for each rasterized fragment. This burden depends on the number of texture stages supported by the hardware. In order to avoid multiple passes for the final scene-shadows rendering, we have utilized the so called *pixel shader* functionality available on programmable GPUs. By using the following hardware-accelerated procedure, only a single rendering pass is required to synthesize shadow effects for all pixels.

For each rasterized fragment, we sample the appropriate shadow map based on the depth value of this fragment. In the view space, obviously T_i should be used for points located in the range $[C_{i-1}, C_i]$. However, in the fragment buffer, the coordinates are measured in the clip space. In Direct3D, the depth value z in the view space is transformed to $\frac{f}{f-n}(1-\frac{n}{z})$ in the clip space. There-

Figure 6: Testing Scenes

Scene	Triangles	TSM (1024x1024)		PSSM(2; 512x512)		PSSM(3; 512x512)		PSSM(4; 512x512)	
		#FPS	#Tex	#FPS	#Tex	#FPS	#Tex	#FPS	#Tex
Robot War	240K	46.40	1Kx1K	32.84	512x512x2	28.40	512x512x3	29.93	512x512x4
Huge Chessboard	51K	67.29		58.00		50.08		47.22	
Knights	1M	33.20		23.20		19.60		18.00	
Mining Terrain	1.3M	22.00		15.60		14.00		13.20	

Table 1: From left to right are testing scenes, the number of triangles in each scene, rendering speed (#FPS: frames per second) and texture memory used (#Tex) for TSM, PSSM(2), PSSM(3) and PSSM(4) respectively.

fore, C_i is transformed to C_i^{clip} below:

$$C_i^{clip} = \frac{f}{f-n}\left(n - \frac{1}{C_i}\right) \in [0,1]$$

Hence, for a pixel in fragment buffer with the depth value z^{clip}, if $z^{clip} \in [C_{index-1}^{clip}, C_{index}^{clip}]$ or

$$\frac{f}{f-n}\left(n - \frac{1}{C_{index-1}}\right) \leq z^{clip} \leq \frac{f}{f-n}\left(n - \frac{1}{C_{index}}\right) \qquad (8)$$

the shadow map T_{index} is selected. Consequently, this fragment is transformed into the split light's frustum W_{index} for the depth comparison.

This step can be simply and purely implemented on the GPU side, by using shader programming languages such as HLSL (High Level Shading Language, Microsoft Corp.), GLSL (OpenGL Shading Language, OpenGL Architectural Review Board) and Cg (C for Graphics, NVidia Corp.). In our system, we implement this procedure by using HLSL on DirectX-9 level hardware. More implementation details can be found at our project webpage.

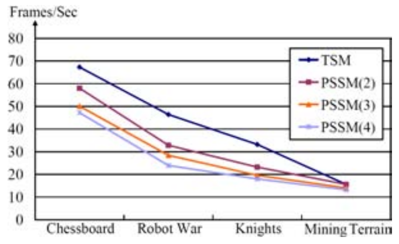
4 Experimental Results

We have developed the Parallel-Split Shadow Maps by using Microsoft DirectX SDK 9.0. We run the shadow rendering tests using 800*600 image resolution, and fps is measured by an Intel Pentium 4 2.8GHz CPU with 1G RAM, a NVidia GeForce 6800Ultra GPU with 256M video memory. The vertex/pixel shader programs are coded to address the rendering of shadow maps and scene-shadows synthesization. In addition, we have compared our PSSMs with single-map shadow mapping techniques including standard shadow maps (SSMs), perspective shadow maps (PSMs), lighting space perspective shadow maps (LiSPSMs) and trapezoid shadow maps (TSMs) from NVidia SDK 9.1. In the experiments of our PSSMs scheme, TSM is selected to generate the shadow maps for the split parts.

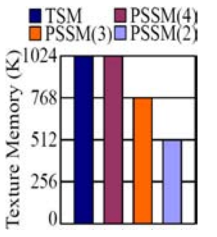
Four virtual scenes are composed and tested in our experiments: **Robot War** (220K triangles), **Huge Chessboard** (51K triangles), **Knights** (1M triangles) and **Mining Terrain** (1.3M triangles), in which a dynamic directional lighting source and field of view 60° are set up (Figure 6). In **Robot War**, there are about 160 static robots randomly located in the $1.44km^2$ environment. 32 static chessmen are placed on a $0.49km^2$ huge chessboard in **Huge Chessboard**. $near = 1m$ and $far = 800m$ are configured for both of them. **Knights** is a virtual desert environment which consists of 60 shadow-casting objects randomly located over a $1.6km^2$ terrain with $near = 1m$ and $far = 800m$. The objects in this virtual scene include 15 dynamic dwarfs, 15 dynamic knights, 15 static trees and 15 static cactus. **Mining Terrain** is a virtual outdoor environment rendered in the view frustum with $near = 1m$ and $far = 1000m$. There are about 55 complex objects including 10 dynamic clawbots, 15 dynamic windmills, 20 static rocks and 10 static trees randomly located in the $2.56km^2$ environment.

The objective of our experiments is to examine how the shadow qualities are improved in PSSMs when we increase the split number of the view frustum. To achieve real-time performance, $m \leq 4$ is adopted in our experiments. This restriction is reasonable because applications need computer resources to produce other realistic rendering effects. Other techniques including SSM, PSM, LiSPSM and TSM are also tested for reference. For each scene snapshot in Figure 8, we have marked and enlarged a rectangle area to show the detail of scene-shadowing qualities. Snapshots for **Huge Chessboard**, **Robot War**, **Mining Terrain**, and **Knights** are listed in the columns (a) to (d) respectively. The rows from top to bottom in each column show the shadows rendered by SSM(1024x1024), PSM(1024x1024), LiSPSM(1024x1024), TSM(1024x1024), PSSM(2; 512x512), PSSM(3; 512x512) and PSSM(4; 512x512) respectively. As we see, the shadow qualities are dramatically increased as the view frustum is split into more depth layers. Figure 8(e) shows the dueling frusta case when the lighting and eye sight are nearly opposite. A single huge shadow map (2048x2048) is used for single-map techniques (SSM, PSM, LiSPSM, TSM). PSSM(2;1024x1024), PSSM(3;1024x1024) and PSSM(4;1024x1024) are used for comparison. In this extreme case, the shadow aliasing shown in single-map shadow mapping techniques increases sharply, even the shadow map resolution is 2048x2048. The results also show that our approach can produce better shadowing qualities in this extreme case.

Table 1 shows the results of timing performance and texture mem-



(a) Rendering Speed (fps)



(b) Texture Memory (K)

Figure 7: Performance Statistics.

ory requirement for single-map shadow mapping technique TSM and PSSMs schemes ($m \leq 4$) used in Figure 8 (a) to (d). The rendering speeds for single-map shadow mapping techniques (SSM, PSM, LiSPSM, TSM) are nearly the same, we thus use TSM as the single-map representative technique to compare with our approach. For instance, the average fps for the **Mining Terrain** scene is 21.9 and standard derivation is 0.8. Table 1 shows that the PSSMs achieved real-time performance (**#FPS** column) for the four virtual scenes. In our experiments, we did not use other optimizations such as LOD management and geometry instancing. The frame rates of PSSM(2) in **Chessboard** and **Mining Terrain** achieve 58.00 fps and 15.60 fps respectively. The frame rates of PSSM(4) in these two testing scenes are 47.22 fps and 13.20 fps. Although a moderate performance hit caused by using multiple shadow maps (multiple rendering passes needed when generating PSSMs), the achieved frame rates for our split scheme is satisfactory. Other optimization techniques may be applied to further improve the performance such as using large batches of geometry if possible. In Figure 7, (a) visualizes the timing performance listed in Table 1, and (b) shows the texture memory used in TSM and our PSSM schemes (**#Tex** column in Table 1). As we see, **#Tex** for PSSMs is usually less than single-map technique. Particularly, our experiments show that PSSM(3) achieves the best tradeoff between the performance and shadow qualities for most practical applications.

In summary, PSSMs can produce real-time shadowing effects with less texture memory, but significant improvement of shadow qualities, especially for the dynamic and large-scale virtual environments. See more dynamic shadowing effects at our project webpage `http://www.cse.cuhk.edu.hk/~fzhang/pssm_vrcia`.

5 Summary

This paper developed the Parallel-Split Shadow Maps (PSSMs) scheme, which splits the view frustum into different depth ranges by using split planes parallel to the view plane, and then renders multiple shadow maps for the split parts. Our approach proposed a fast and robust split scheme without expensive scene analysis per frame, which produces moderate sampling densities over the whole depth range. The single shadow map with a large size in other shadow mapping methods is replaced by multiple shadow maps with smaller sizes in our approach, the total texture memory usage in PSSMs is usually less than that in single-map methods but better scene-shadows qualities. Our experimental results have shown the high-quality shadow rendering on the commodity graphics hardware with real-time performance, especially for the dynamic and large-scale virtual environments.

We will further our work on hardware-accelerated PSSMs to reduce rendering passes for the generation of shadow maps (e.g. using MRT on current shader model), and the formal analysis of perspective aliasing distribution in PSSMs. The effective integration of shadow mapping techniques in our parallel-split scheme would also be investigated.

Acknowledgment

This work was supported by RGC research grant (ref. 4181/03E), UGC direct grant for research (no. 4450002). We would like to thank Jane for the helpful work on the illustrative figures, Chenjun Tao for preparing experimental equipments and paper formatting. Thanks to Fei Wu, Hao Shen, Yong He, Guangyi Shi, Mei Dong and Ming Dong for valuable discussions and suggestions. The models used in experiments are downloaded from *www.planetquake.com*, *www.3dcafe.com* and *developer.nvidia.com*.

References

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

BRABEC, S., ANNEN, T., AND SEIDEL, H.-P. 2002. Practical shadow mapping. *J. Graph. Tools 7*, 4, 9–18.

CHONG, H. Y., AND GORTLER, S. J. 2004. A lixel for every pixel. In *the Eurographics Symposium on Rendering 2004*, Eurographics, Eurographics Association.

CROW, F. C. 1977. Shadow algorithms for computer graphics. In *Proceedings of SIGGRAPH '77*, ACM Press, 242–248.

FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. 2001. Adaptive shadow maps. In *Proceedings of SIGGRAPH '01*, ACM Press, 387–390.

GOVINDARAJU, N. K., LLOYD, B., YOON, S.-E., SUD, A., AND MANOCHA, D. 2003. Interactive shadow generation in complex environments. *ACM Trans. Graph. 22*, 3, 501–510.

KOZLOV, S. 2004. Perspective shadow maps: care and feeding. *GPU Gems*, Addison–Wesley.

LEFOHN, A., SENGUPTA, S., KNISS, J. M., STRZODKA, R., AND OWENS, J. D. 2005. Dynamic adaptive shadow maps on graphics hardware. In *ACM SIGGRAPH 2005 Conference Abstracts and Applications*.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *Proceedings of SIGGRAPH '00*, ACM Press, 385–392.

LOW, K.-L., AND ILIE, A. 2003. Computing a view frustum to maximize an object's image area. *J. Graph. Tools 8*, 1, 3–15.

MARTIN, T., AND TAN, T.-S. 2004. Anti-aliasing and continuity with trapezoidal shadow maps. In *the Eurographics Symposium on Rendering 2004*, Eurographics, Eurographics Association.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. In *Proceedings of SIGGRAPH '87*, ACM Press, 283–291.

SEN, P., CAMMARANO, M., AND HANRAHAN, P. 2003. Shadow silhouette maps. *ACM Trans. Graph. 22*, 3, 521–526.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. In *Proceedings of SIGGRAPH '02*, ACM Press, 557–562.

TADAMURA, K., QIN, X., JIAO, G., AND NAKAMAE, E. 2001. Rendering optimal solar shadows with plural sunlight depth buffers. *The Visual Computer 17*, 2, 76–90.

WANG, Y., AND MOLNAR, S. 1994. Second-depth shadow mapping. Tech. rep., University of North Carolina at Chapel Hill.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH '78*, ACM Press, 270–274.

WIMMER, M., SCHERZER, D., AND PURGATHOFER, W. 2004. Light space perspective shadow maps. In *the Eurographics Symposium on Rendering 2004*, Eurographics, Eurographics Association.

WOO, A., POULIN, P., AND FOURNIER, A. 1990. A survey of shadow algorithms. *IEEE Comput. Graph. Appl. 10*, 6, 13–32.

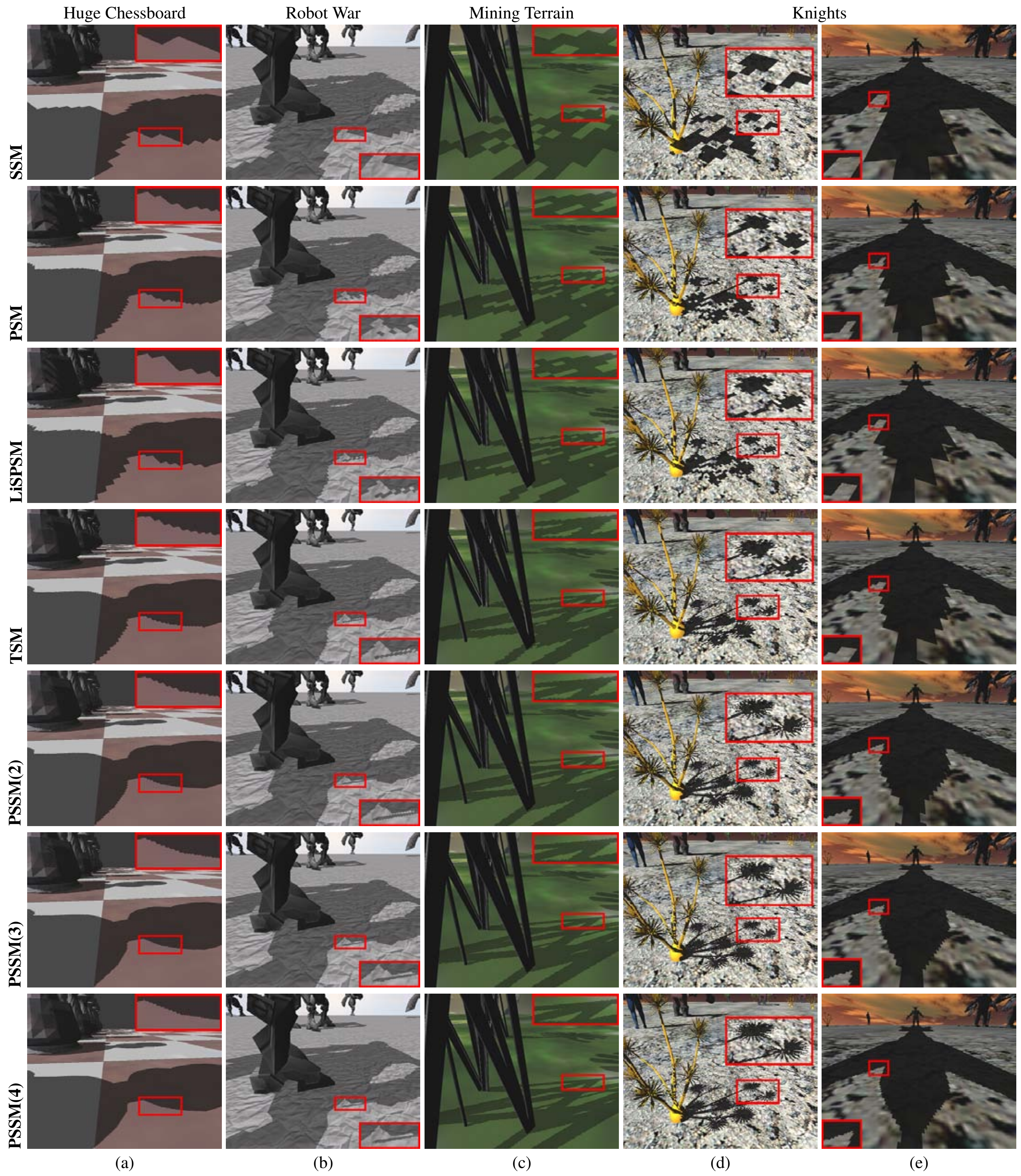


Figure 8: Columns (a)∼(d): snapshots for **Huge Chessboard**, **Robot War**, **Mining Terrain**, **Knights**. The rows from top to bottom in each column between (a)∼(d): shadows rendered by SSM(1024x1024), PSM(1024x1024), LiSPSM(1024x1024), TSM(1024x1024), PSSM(2; 512x512), PSSM(3; 512x512) and PSSM(4; 512x512) respectively. Column (e): the dueling frusta case (i.e. the light direction is nearly opposite to the view direction) for **Knights**. The rows from top to bottom in (e): shadows rendered by SSM(2048x2048), PSM(2048x2048), LiSPSM(2048x2048), TSM(2048x2048), PSSM(2; 1024x1024), PSSM(3; 1024x1024) and PSSM(4; 1024x1024) respectively.