High-Quality Shadows with Improved Paraboloid Mapping

Juraj Vanek, Jan Navrátil, Adam Herout, and Pavel Zemčík

Brno University of Technology, Faculty of Information Technology, Czech Republic
http://www.fit.vutbr.cz

Abstract. This paper introduces a novel approach in the rendering of high-quality shadows in real-time. The contemporary approach to shadow rendering in real-time is shadow maps algorithm. The standard shadow maps suffer from discretization and aliasing due to the limited resolution of the shadow texture. Moreover, omnidirectional lights need additional treatment to work correctly (e.g. with dual-paraboloid shadow maps). We propose a new technique that significantly reduces the aliasing problem and works correctly for various kinds of light sources. Our technique adapts the light's field-of-view to the visible part of the scene and thus stores only the relevant part of the scene to the full resolution of the shadow map. It produces high-quality shadows which are not constrained by position or type of the light sources and also works in fully dynamic scenes. This paper describes the new approach, presents some results, and discusses the potential, features and future of the approach.

1 Introduction

Shadows constitute an important part of computer graphics rendering methods because they allow for better perception of the depth complexity of the scene. Without shadows human viewers are not able to fully determine in-depth object positions in the virtually created scene. Virtual scenes are often very dynamic, so if we want to achieve high quality shadows the algorithm has to be robust and

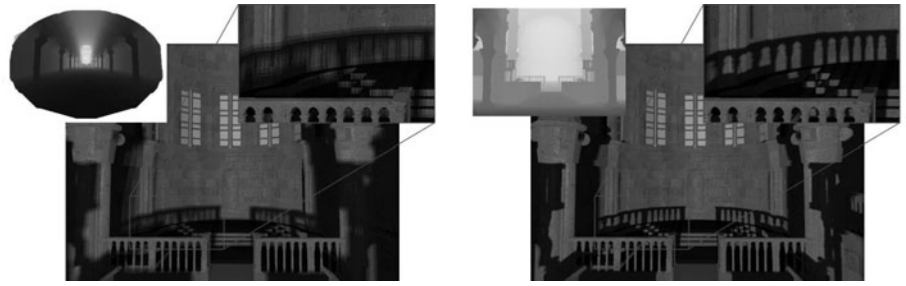

Fig. 1. Large indoor scene with hemispherical light source and 1024×1024 paraboloid shadow map (Left). Same scene with improved paraboloid map of the same size (Right).

work regardless of the scene's configuration, specifically the light and camera position. An example can be seen in applications for modeling and visualization where developers require fast and accurate shadow casting, independent from light types, camera position and scene complexity.

At present, the shadow algorithms used in real-time applications can be divided into two large groups. The first one is based on shadow volumes [1] and the second one on shadow maps [2]. Each of these two methods has advantages and disadvantages and is differently suitable for different types of light sources. Shadow volumes can be easily used to implement omnidirectional lights with per-pixel accuracy without any modifications. However, they depend heavily on scene geometry so in complex scenes this approach is very fill-rate intensive and can hardly ever be used to compute shadows in such scenes in real-time. On the contrary, shadow maps algorithm is much less dependent on the geometric complexity of the scene as it only requires a single additional drawing pass into the depth buffer. However, being a discrete image-space algorithm, the shadow maps algorithm suffers from discretization and aliasing problems and, moreover, the implementation of omnidirectional shadows from point lights is not straightforward because it requires more than one shadow map [3]. At present time, with increasing scene's geometrical complexity, the shadow maps algorithm is the most frequently used one to render shadows in real-time applications despite its limitations.

Our approach introduces a new method that improves the dual-paraboloid shadow maps method first introduced by Brabec et al [3] and further enhanced by Osman et al. [4]. This improvement is based on adding optimal paraboloid orientation and cutting the paraboloid unevenly to improve the sampling of shadows in important areas corresponding to the view frustum. It allows for the rendering of shadows with high quality regardless of light type (e.g point light, spot light or directional light) and light or camera position. Therefore, the method is suitable for large, dynamic, indoor and also outdoor scenes where high accuracy shadows are needed, but because of the high geometric complexity of the scene, the traditional approaches to render shadows from point lights like the shadow volumes cannot be used. Our solution is designed to run optimally on modern graphics accelerators and the measurements show no measurable performance penalty compared to the original dual-paraboloid shadow maps.

2 Previous Work

As mentioned above, two main branches of shadow algorithms are contemporarily being used for real-time rendering. A shadow volumes algorithm [1] works in the object space and a shadow maps algorithm [2] works in the image space.

Shadow volumes are created from an object's silhouettes towards the light source. Since this algorithm works in the object space, all shadows are hard with per-pixel accuracy and they are very suitable and simply applicable for omnidirectional lights. However, a generation of the silhouette and rendering of the shadow volumes is done on the CPU, so for scenes with high geometry complexity, the algorithm is slow. However, modifications allowing for acceleration of computation of the silhouettes on the GPU are available [5].

The shadow maps algorithm presents an advantage compared to the shadow volumes algorithm because it is not as dependent on the scene complexity. However, as the shadow texture has limited resolution, sampling and aliasing problems inevitably occur. Nowadays, when geometrical complexity of graphics models and scenes is increasing, the shadow maps algorithm seems to be the better solution for real-time scene shadowing despite its limitations.

Many methods of removing shadow map imperfections exist, especially the ones dealing with the sampling and aliasing problems [6][7][8].

In these methods, the scene and the light source are transformed into the post-perspective space and the shadow map is generated in this space as well. The Cascaded Shadow Maps algorithm [9] can be seen as discretization of these approaches where re-parameterization is replaced with the use of multiple shadow maps.

The above mentioned shadow mapping techniques work when used with spot lights only because when generating the shadow map, the field of view must be limited. In order to obtain shadows from omnidirectional light sources, the whole environment needs to be covered as seen from the light source. At the moment, multiple shadow maps [3,4] seem to be the only available solution. The most straightforward solution is to render the scene into six shadow maps for each side of the cube so all scene directions are captured. This process requires no additional scene warping and hardware perspective projection with cube mapping can be used, but six additional passes to render the scene present a serious disadvantage. Although it is possible to use single-pass rendering with a geometry shader on the newest hardware, GPU still has to create six additional vertex render passes, making it unsuitable for complex, dynamic scenes.

A dual-paraboloid shadow maps algorithm (DPSM), first introduced by Brabec et al. [3], is in most cases a better solution. It uses only two passes and the scene is rendered in a 180° field of view for each pass, while paraboloid projection is used for each vertex. Paraboloid projection does not suffer from scene distortion and singularities that occur in traditional sphere mapping: the two paraboloids are put back to back, covering full a 360° view of the environment around the light source. Since depth values are stored according to the paraboloid scene representation in the texture, texture coordinates need to be computed appropriately. The original DPSM algorithm suffers from a tessellation problem as only the vertices are projected by the paraboloid projection, but the entities are rasterized linearly.

Most of such issues can be solved through the approach shown by Osman et al. [4]. The tessellation problem for shadow receivers can be solved by computing the paraboloid shadow map coordinates in a per-pixel manner and the shadow casters can be adaptively tessellated based on triangle size and distance from the light source using a hardware tessellation unit.

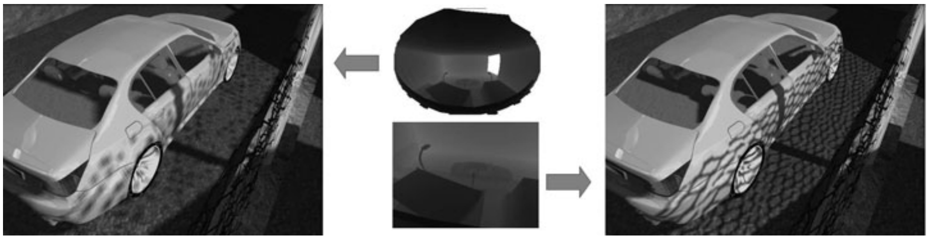

Fig. 2. Small outdoor scene with hemispherical light source and 1024×1024 paraboloid shadow map (Left). Same scene with improved paraboloid map of the same size (Right). One should note the improvement of the shadow through a transparent fence texture.

3 High-Quality Shadows from Point Light Sources

Our solution extends the idea of the dual-paraboloid shadow maps (DPSM) [3] combined with some improvements proposed by Osman et al. [4], specifically the proper choice of the splitting plane, tessellation of shadow casters, and per-pixel computation of paraboloid coordinates. However, one serious drawback of the original dual-paraboloid shadow maps still remains. A DPSM algorithm is based on a uniform distribution of shadow samples from the paraboloid resulting in sampling and aliasing artifacts, especially when the viewer is far from the light position. In the following section, we will present a solution which can notably reduce the aliasing artifacts.

The proposed idea is to adjust the sampling density on the paraboloid in order to increase sampling density in the parts of the scene that are close to the viewer. The approach is based on the fact that it is not necessary to sample the whole paraboloid. Moreover, some parts of the paraboloid can be completely excluded from sampling in the shadow texture. Also, there is no need for expensive two-pass rendering into both paraboloids when the light source lies outside the view frustum and the single paraboloid is sufficient to cover the whole view frustum with a single shadow map. However, this approach works only when the light lies outside the view frustum. When light is positioned inside, the algorithm falls into the regular DPSM.

3.1 Improved Paraboloid Shadow Maps

The original DPSM algorithm uses two paraboloids - front and back - in order to capture the scene environment around the light source from both directions. Each paraboloid is sampled uniformly and the rotation of the paraboloids remains fixed. The environment is captured in two depth textures. Shadow map generation is then done by calculating the paraboloid projection and such projection can be performed by using a vertex shader [3].

Example of the vertex shader performing the paraboloid projection

```
vec4 vertexEyeSpace = in_LightModelViewMatrix * vec4(in_Vertex,1.0);
```

```
vertexEyeSpace.xyz = normalize( vertexEyeSpace.xyz );
vertexEyeSpace.z += 1.0;
vertexEyeSpace.xy /= vertexEyeSpace.z;
```

In order to achieve better sampling of the shadow map, we must ensure that rendering from the light source point of view with parabolic projection will maximally cover the objects lying inside the camera view frustum (it is not necessary to sample the scene parts for objects which receive shadows outside the view frustum). Therefore, the algorithm consists of four main steps:

1. Locate clipping planes for the view frustum to mark the boundary where all potential shadow receivers are present, forming *a truncated view frustum.*
2. Determine an optimal field of view for the scene rendered from the light source to cover the whole truncated view frustum.
3. Find the directional vector which sets the rotation of the paraboloid.
4. Perform a suitable cut on the paraboloid to constrain shadow map rendering only to the selected part of the paraboloid.

These steps lead to an improvement of the paraboloid shadow maps, so we will refer to them as *improved paraboloid shadow maps* (IPSM) in the following text. Because rotating of the front paraboloid will cover most of the view frustum, the back paraboloid is needed only when the light lies inside the view frustum. In such cases, shadows must cover the entire environment around the view frustum and the back paraboloid is needed. In such cases, the parameterization converges to the standard DPSM. Otherwise, we can save one rendering pass and speed up rendering.

3.2 Determining Optimal Coverage

In the 3D space, an area illuminated by a point light with certain field-of-view and direction has the shape of a cone, as seen in Figure 3 (right). We will refer to it as *a light cone* below. In order to optimally cover the view frustum with the light cone, we need to locate the boundaries around the visible objects in the view frustum first, so that shadow sampling will be performed only within its boundaries. We will refer to them as *minimal depth* and *maximal depth clipping planes.* In order to determine the boundaries we store the z-values of all transformed and clipped vertices in the eye space to obtain minimum and maximum z-values. In the next text, we will refer to the view frustum with maximum/minimum depth boundaries as *a truncated view frustum* (see Figure 3 left).

In order to locate optimal coverage and field-of-view, we suggest the following method. Prior to starting the calculation, we obtain a light position L and a position of eight *frustum border points* (FBPs) on the minimal ($N_{1..4}$) and maximal ($F_{1..4}$) depth clipping planes of truncated view frustum, as seen in Figure 3 (left). The algorithm computes the optimal field-of-view F_v and direction vector for the light cone C_d.

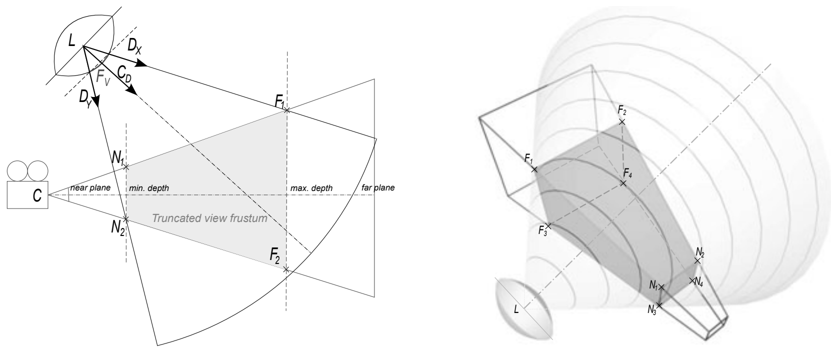

Fig. 3. Illustration of optimal coverage of the truncated view frustum with the light cone

Firstly, the normalized sum of the vectors D_j from the light source to the FBPs is computed (Eq. 1). This computation expresses the average direction $\bar{C_d}$ from the light source L to the truncated view frustum.

$$\bar{C_d} = norm(\sum_j D_j) \tag{1}$$

In the next step, we iterate through all vectors D_j in order to obtain the maximal angle from FBPs to an average direction $\bar{C_d}$ (Eq. 2).

$$F_v = max(D_j \cdot \bar{C_d}), 1 \leq j \leq 8 \tag{2}$$

This maximal angle defines the field-of-view F_v of the light cone. Since all frustum border points are contained in the light cone, this implies that the whole truncated view frustum will also be covered by the light cone (Figure 3 right). This evaluation is performed only once per frame, so it does not have a crucial impact on the performance. It should be noted that we propose the numerical solution and thus it is not optimal.

3.3 Paraboloid Cutting and Rotation

After obtaining the field of view F_v and the average direction vector $\bar{C_d}$ using the steps above, the light cone can be defined to cover the truncated view frustum. Since the single light paraboloid has a 180° field-of-view, we suggest the cutting scheme that is used for smaller angles.

We exploit the parameterization for DPSM (see Section 3.1) to find the zoom factor. We set an auxiliary vector whose direction is derived from the field-of-view F_v. Then we let the vector be processed in the same manner as the transformed vertices position (*vertexEyeSpace*) in the vertex shader. This processing could be done in 2D because of the symmetry of the paraboloid. The resulted x-axis coordinate expresses the zoom factor Z which is applied to the computed coordinates in the vertex shader: $vertexEyeSpace.xy* = 1/Z$. RThis operation

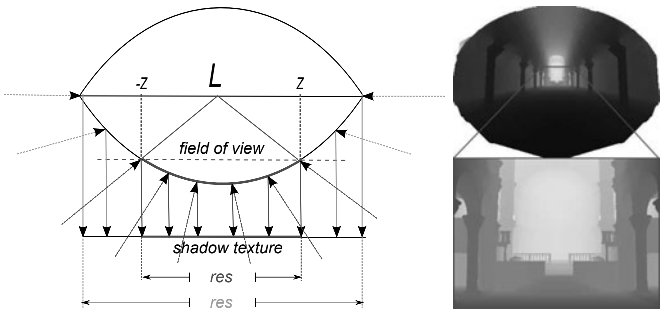

Fig. 4. Paraboloid cut to constrain shadow map sampling to certain parts of the paraboloid

causes the paraboloid to be cut at the point $(Z, f(x))$ which precisely creates the desired field-of-view F_v (see Figure 4), but the resolution of the shadow map texture on the output remains the same.

4 Implementation

The implementation was done using an OpenGL 4 core profile in our own framework with use of the GLSL shading language. The scenes were rendered in the 1600×1200 resolution on the PC with Intel Core i5 3,33GHz, 4GB RAM, AMD Radeon 5970 and Windows 7 32-bit. We used three demonstration scenes: Sibenik cathedral (70k of polygons), car (160k of polygons) and trees (150k of polygons). The purpose of the scenes was to demonstrate various light source types (see Figure 5).

The shadow maps for the front and back paraboloids are contained in a texture array for easy accessing and the indexing of both textures. In order to avoid artifacts when rendering into the shadow map with parabolic projection on sparse tessellated mesh, a hardware tessellation unit is used. Tessellation factor is based on the distance of the tessellated polygon near the edge of paraboloid where possible distortion can occur. New vertices generated by the tessellator are parabolically projected in the tessellation evaluation shader.

In order to compute minimal/maximal depth clipping planes we need z-values of the transformed vertices, so the scene's normal and depth values are rendered into a floating point render target with the attached texture. This process is common in graphics engines using deferred shading and can be easily added into the scene rendering pipeline.

This texture is scaled down on the GPU to a small size (e.g. 32^2) so as to reduce data transfer (we do not need per-pixel accuracy) and is then transferred from GPU to CPU. The stored buffer is analyzed and minimum, maximum and average depth values are searched for. Because of the scaling on the GPU and the transfering of only a small block of data to CPU, the effect on performance is not very big (as will be seen in the next section).

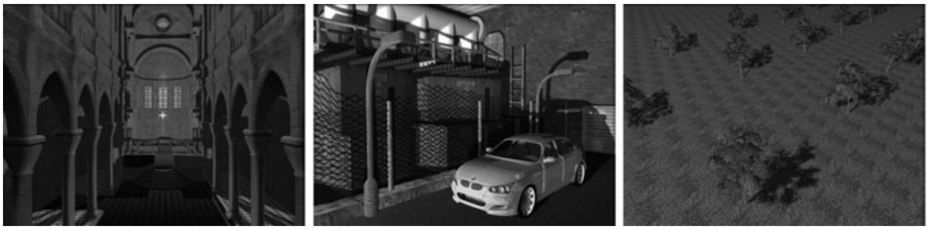

Fig. 5. Examples of various light source types with IPSM. From the left: omnidirectional, hemispherical and directional.

Shaders from the original DPSM can be left almost intact and our solution can be easily implemented into any graphics engine using deferred shading techniques without major changes against the original DPSM.

5 Experimental Results and Discussion

As can be seen in Figures 1, 2, 6 and 7, improved paraboloid shadow maps (IPSM) can significantly improve the visual look of the shadow in the scene without the necessity of large shadow maps. For all of our test scenes, resolution 1024×1024 was sufficient. When the light lies outside the view frustum, rotating and cutting the paraboloid produces far better results than the original DPSM or the uniform shadow maps (Figures 1, 2 and 7) because of optimal paraboloid rotation covering all objects lying inside the view frustum. Another interesting fact is that closer the camera gets to the shadow of the object, the more detailed the shadow will appear as a result of small differences between minimal and maximal depth clipping planes. This is a result of dense sampling of this small part of view frustum (Figure 2).

It is also possible to use IPSM in outdoor scenes (Figure 7.) with distant light which acts in this case as directional light.

However, when light lies inside the view frustum, only paraboloid rotation can be used because we cannot cut the paraboloid as we could lose important shadow information. Therefore, in this situation the field-of-view is always 180° and our solution is equal to the original DPSM (Figure 5, left). The worst case for our algorithm is, therefore, a camera near to the ground looking at the distant light and an object casting shadow towards camera - such a shadow will have poor quality, but this is also an issue of the original DPSM.

Although additional calculations when compared to the original DPSM need to be performed, all of them are calculated per-frame only, so the performance impact is very low and close to the border of measurement error, as can be seen in Table 1.

When compared to the uniform shadow maps, DPSM and IPSM are slower because of the higher computational complexity of parabolic projection. On the other hand, we cannot use a single uniform shadow map to capture the whole

Table 1. Performance of the different shadow approaches (in frames per second). Measured on the all demonstration scenes with AMD Radeon 5970.

	Sibenik		Car		Trees	
Shadow resolution	512^2	2048^2	512^2	2048^2	512^2	2048^2
Uniform SM	180	140	162	142	217	197
DPSM	102	78	97	84	135	123
IPSM	95	76	107	95	148	134

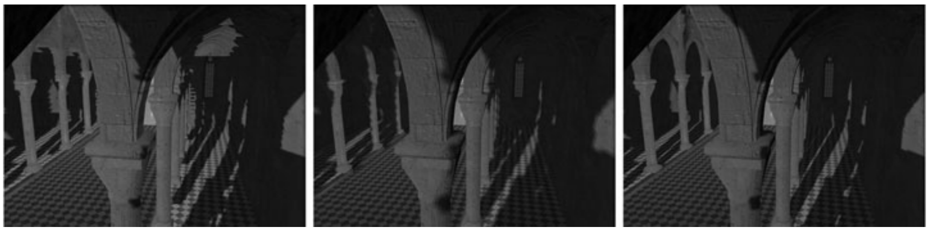

Fig. 6. This image sequence shows differences between shadow methods. From left to right: uniform shadow maps, DPSM and IPSM. The shadow resolution is 512 × 512.

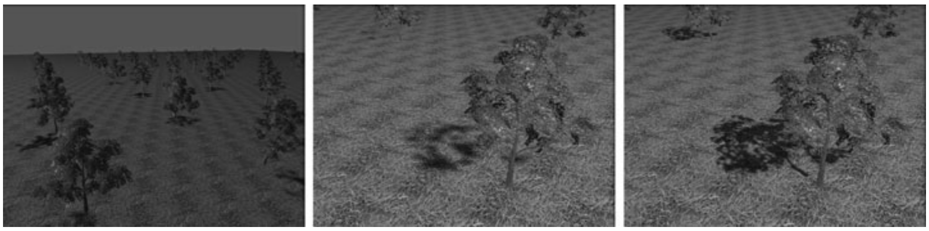

Fig. 7. Example of the large outdoor scene with the uniform shadow maps (Left) with close-up on a single tree (Middle) and IPSM (Right). In this case, paraboloid projection is similar to a focused parallel shadow map and is also suitable for directional lights without any changes. The shadow map resolution is 1024 × 1024.

environment. IPSM can be in some cases faster than the original DPSM because when light lies outside the view frustum, back paraboloid can be omitted as front paraboloid can cover all visible objects in the view frustum itself (Figure 3). Otherwise, IPSM are slower due to additional computations (mainly determining minimal and maximal depth values), but this performance hit is not significant.

6 Conclusion

This contribution introduces an improvement to existing paraboloid shadow maps based on the dense sampling of the important areas in the view frustum. Our solution (IPSM) can in most cases provide accurate, high quality shadows regardless of the light source type and its position, without any performance penalty when compared to original paraboloid shadow maps. The solution is suitable for dynamic, indoor as well as outdoor scenes with the high geometric complexity where sharp, high quality shadows are needed with the various light source types.

It is worth mentioning that because IPSM evolved from the standard DPSM, it should be possible to implement our method to any modern graphics engine without too much effort.

Future work should include further improvements, such as solving the situation where the light source lies inside the view frustum and we need effectively sample areas close to the viewer. We also want to exploit the functions of modern graphic accelerators and geometry shaders to render all two shadow textures at once in a single pass, thus speeding up rendering and real light sources simulation by blurring shadows based on the distance from such light sources.

Acknowledgement. This work was supported by the Ministry of Education, Youth and Sport of the Czech Republic under the research program LC06008 (Center of Computer Graphics). The Sibenik cathedral model is courtesy of Marko Dabrovic.

References

1. Crow, F.C.: Shadow algorithms for computer graphics. SIGGRAPH Comput. Graph. 11, 242–248 (1977)
2. Williams, L.: Casting curved shadows on curved surfaces. SIGGRAPH Comput. Graph. 12, 270–274 (1978)
3. Brabec, S., Annen, T., Seidel, H.P.: Shadow mapping for hemispherical and omnidirectional light sources. In: Proceedings of Computer Graphics International, pp. 397–408 (2002)
4. Osman, B., Bukowski, M., McEvoy, C.: Practical implementation of dual paraboloid shadow maps. In: Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames, pp. 103–106. ACM, New York (2006)
5. Stich, M., Wächter, C., Keller, A.: Efficient and Robust Shadow Volumes Using Hierarchical Occlusion Culling and Geometry Shaders. In: GPU Gems 3. Addison-Wesley Professional, Reading (2007)
6. Fernando, R., Fernandez, S., Bala, K., Greenberg, D.P.: Adaptive shadow maps. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Technique, pp. 387–390. ACM, New York (2001)
7. Stamminger, M., Drettakis, G.: Perspective shadow maps. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 557–562. ACM, New York (2002)
8. Wimmer, M., Scherzer, D., Purgathofer, W.: Light space perspective shadow maps. In: The Eurographics Symposium on Rendering (2004)
9. Zhang, F., Sun, H., Xu, L., Lun, L.K.: Parallel-split shadow maps for large-scale virtual environments. In: Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and its Applications, pp. 311–318. ACM, New York (2006)