IEICE TRANS. INF. & SYST., VOL.E100–D, NO.1 JANUARY 2017

LETTER

Light Space Partitioned Shadow Maps

Bin TANG[†], ***Nonmember***, **Jianxin LUO**[†], ***Member***, **Guiqiang NI**[†a)], **Weiwei DUAN**[†], ***and*** **Yi GAO**[†], ***Nonmembers***

SUMMARY This letter proposes a Light Space Partitioned Shadow Maps (LSPSMs) algorithm which implements shadow rendering based on a novel partitioning scheme in light space. In stead of splitting the view frustum like traditional Z-partitioning methods, we split partitions from the projection of refined view frustum in light space. The partitioning scheme is performed dual-directionally while limiting the wasted space. Partitions are created in dynamic number corresponding to the light and view directions. Experiments demonstrate that high quality shadows can be rendered in high efficiency with our algorithm.

key words: *projection plane, dual-directional partitioning, light frusta*

1. Introduction

Shadow rendering is a hot topic in Computer Graphics. It is very important to enhance the realism in games and movies. Shadow maps are one of the most popular techniques for rendering shadows. A two pass shadow map algorithm was firstly proposed by Williams [1], which creates a depth map by rendering the scene in light space firstly, and then renders the scene again in view space to display shadows with the help of depth comparison techniques.

Many researchers have dedicated to the generation of shadows from different aspects [2]. Among these aspects, partitioning is one of the fundamental techniques which partitions the view frustum into splits for generating shadow maps. Determining the number of partitions and viewport of each partition are essential issues for rendering the high quality shadows in high efficiency. In this letter, we concentrate on a novel partitioning algorithm which creates shadow maps more efficient than traditional partitioning algorithms. More precisely, the main contribution of this letter are as follows:

Light Space Partitioning. Shadow maps are created by partitioning the projection the refined view frustum in light space dual-directionally. Overlapping problem is eliminated easily for improve image quality.

Auto-tuned Shadow Maps. Shadow maps are auto-tuned in dynamic number for raising rendering efficiency. Viewport of each shadow map is calculated by limiting the wasted space at best effort, instead of by determining the Z-partition point like traditional methods.

Manuscript received August 18, 2016.
Manuscript publicized October 4, 2016.
[†]The authors are with the PLA University of Science and Technology, Nanjing, P.R. China.
a) E-mail: nigq1966@163.com
DOI: 10.1587/transinf.2016EDL8171

2. Related Works

In this section, we list some influential contributions related to partitioning for shadow maps. Currently, most partitioning algorithms are originated from Cascaded Shadow Maps (CSMs) algorithm [3], which splits the view frustum in Z-direction and creates shadow maps based on the partitioned sub-frusta.

A detailed discussion about determining the splitting point (Z_i in Fig. 1) is proposed in PSMs (Parallel-Split shadow Maps) [4]. Lloyd [5] discussed low error shadow maps by combing the partitioning with warping techniques. However, such schemes are sensitive to a hand tuned parameter, which influences the shadow quality and must be set depending on the scene rendered. To create partitions automatically, Lauritzen [6] proposed a Sample Distribution Shadow Maps (SDSMs) which determines the splitting point by analyzing the distribution of the light space samples. This method can raising the rendering efficiency in local scenes, like street or indoor scenes. However, another problem of viewports overlapping between shadow maps frusta still exists, which wastes shadow map space and increases average frame time. Liang [7] proposed a Light Space Cascaded Shadow Maps (LiSCSMs) algorithm which prevents a same geometry casting into multiple shadow maps for improving rendering efficiency, but the space overlapping problem still exists. Nikolas [8] proposed a "Oblique Projection for Cascaded Shadow Maps (OPCSMs)" method to eliminate the overlapping problem in their "CryENGINE". However, this method is still based on Z-partitions of view frustum, and light frusta are created based on the partitions in 3 view frustum clip planes for wholly covering the view frustum. CryENGINE consumes 2 times more rendering passes than traditional CSMs algorithms [3],

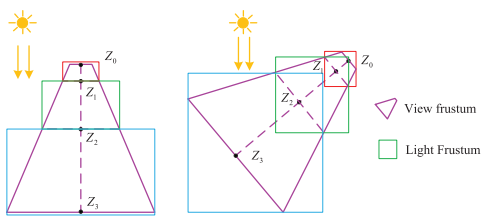

Fig. 1 Large proportion of view ports of light frusta overlap with each other in tradition partitioning algorithms. Left: light direction is parallel with view direction. Right: light is in an arbitrary angle to view direction.

showing a poorer performance in rendering efficiency. Besides, current partitioning methods, to our knowledge, create partitions in fixed number in whatever light and view directions. The fixed partitions make it difficult to raise higher efficiency for traditional Z-partitioning based methods.

3. Light Space Partitioning

We describe Light Spaced Partitioned Shadow Maps (LSPSMs) algorithm in this section, whose core principle is creating non-overlapped partitions for shadow maps in light space. Partitions are generated in dynamic number corresponding to the light and view directions. Our goal is raising the rendering efficiency while keeping high image quality. The LSPSMs is simple enough to add to existing shadow maps systems easily.

3.1 Light Projection Plane

We introduce a terminology "light projection plane", which refers to a virtual plane perpendicular to the light direction, as Fig. 2 shows. Instead of splitting the view frustum directly like traditional partitioning based algorithms [2], [3], we split the projection of the view frustum in light projection plane.

Before projecting, the view frustum can be refined to bound the shadow-viewed space tightly by removing the space below the base plane, as well as the space without any shadow object inside. We name the projection of the refined view frustum in light space as *the projective region*, which is in a shape of isosceles trapezoid. Partitions are then split from the projective region according to a dual-directional partitioning scheme, which is detailedly discussed in Sect. 3.2. Finally, shadow maps are generated by the light frusta, whose viewports are defined by these partitions.

In the perspective of light, the shadow viewed space inside the view frustum is wholly covered by the projective region. Non-overlapped partitions can be directly split from the projective region to create light frusta. Overlapping problem is easily eliminated in this way. Comparing to CryENGINE, which creates partitions from 3 clip planes, our LSPSMs implements partitioning operations only in one plane. This novelty make LSPSMs owning the potential to create shadow maps with high sample density in more efficient way.

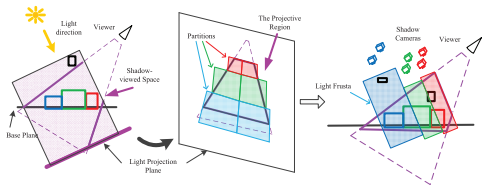

Fig. 2 An overview of of LSPSMs. Left: shadow viewed space is projected into light space to create the projective region. Middle: partitions are created based on the projective region in two directions. Right: light frusta are created based on the partitions for shadow maps generating.

3.2 Dual-Directional Partitioning

Our dual-directional partitioning strategy performs based on shape of the projective region, which indicates the sample distribution in reverse proportion to the size. As Fig. 3 shows, n and f denote the length of near and far base respectively. L is the length of the projective region in view direction. θ ($0 \le \theta < \pi/2$) is the angle between view direction and a leg of the isosceles trapezoid-shaped projective region. Their values vary corresponding to two factors, including view and light directions. In the case of both view and light directions being orthogonal to the ground, the projective region expands to a rectangle, which tells $\theta = 0$ and $n = f$. Samples inside the viewport own close footprints in this case. In other cases, footprints of samples around the near side are smaller than the farther samples.

There are two steps to split the projective region, including lengthways partitioning in first, and transverse partitioning in second.

Lengthways partitioning intends to create partitions in rectangle shape. For creating shadow maps with high sample density, we determine the sides of each partition by limiting the the maximum waste space at best effort, rather than creating splitting points in Z-direction like traditional methods [3], [8].

Let's denote the orthogonal side and lengthways side of a partition as W_i and H_i respectively (i is the index of a partition starting from 1). The space wasting factor of partition i is calculated as $\rho = \frac{H_i * tan\theta}{W_i}$. ζ is introduced to represent the maximum space wasting tolerance, which means $\rho < \zeta$ ($0 \le \zeta < 0.5$). Based on this limitation, we compute the sides of each partition as

$$W_i = \begin{cases} \left(\frac{1}{1-2\cdot\tan\theta}\right)^i \cdot n & \theta \in [0,\ \arctan\zeta) \\ \left(\frac{1}{1-2\cdot\zeta}\right)^i \cdot n & \theta \in [\arctan\zeta,\ \pi/2) \end{cases}$$
$$H_i = \begin{cases} \left(\frac{1}{1-2\cdot\tan\theta}\right)^i \cdot n & \theta \in [0,\ \arctan\zeta) \\ \left(\frac{1}{1-2\cdot\zeta}\right)^i \cdot \frac{\zeta}{\tan\theta} \cdot n & \theta \in [\arctan\zeta,\ \pi/2) \end{cases} \quad (1)$$

Equation (1) shows a scheme to create partitions heuristically from the near side of the projective region. It can be easily proved that, in any case, the space wasting factor is limited as $\rho \le \zeta$.

The lengthways partitioning accomplishes once the whole projective region is covered by the partitions when the partition number h_{max} meets $\sum_{n=1}^{h_{max}} H_i \ge L$. From Eq. (1) we know that the value of H_i is only subject to θ and ζ. The value of ζ is set by user, which is a constant parameter in all

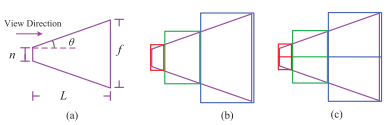

Fig. 3 Dual-directional partitioning. (a) The projective region. (b) Partitions are created in lengthways direction. (c) Transverse partitioning.

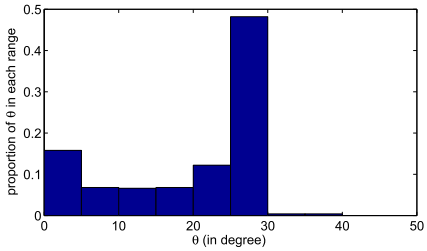

Fig. 4 Distribution of θ during viewer wandering.

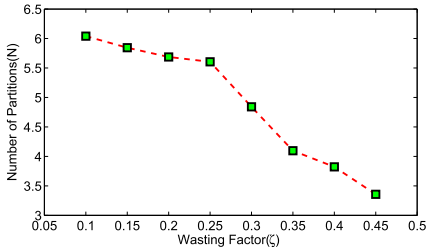

Fig. 5 Average partition number (N) to the wasting factor ζ.

frames in fact. So, the number of lengthways partitions is tuned by θ automatically corresponding to the changing of light and view directions in each frame.

One of the advantages of the varied partition number is that the rendering efficiency can be raised due to decreased average rendering pass. This feature makes our partitioning scheme different from all existed partitioning algorithms in fixed partition number and facilitates the high quality shadow rendering in energy sensitive devices.

Transverse partitioning is performed after the lengthways partitioning. Usually, shadow maps are created in resolution of $N \times N$. uv footprint difference would be introduced by the different length of viewport sides. To improve the image quality by balancing the uv footprints, we introduce a maximum footprint difference tolerance factor σ, which represents the upper limit of the uv footprint difference as $\frac{W_i}{H_i} \leq (1+\sigma)$ $(\sigma > 0)$. Once a lengthways partition exceeds the uv footprint difference tolerance as $\frac{W_i}{H_i} > (1+\sigma)$, an transverse partitioning is performed to split it into N_w sub-partitions $(N_w = \left\lceil \frac{W_i/H_i}{1+\sigma} \right\rceil)$. Transverse side length of newly split partitions are calculated as $W_{ij} = \frac{W_i}{N_w}$.

4. Results

We have implemented Light Space Partitioned Shadow Maps (LSPSMs) on a desktop computer equipped with Core I7 CPU, 8G memory, and GTX 780 GPU. All the shadow maps are set in size of 1024×1024. The screen viewport is 1440×1080. The field of view is 60 degree and the aspect ratio of the view frustum is 1.33 (1440/1080).

4.1 Performance

From Eq. (1) we know that the partition number in lengthways direction is influenced by two parameters including ζ and θ, the former is defined by user and the latter is auto-tuned during the viewer wander through the scene.

Figure 4 is a histogram which shows the distribution of θ during viewer wandering through an arbitrary pass. The motion of the viewer includes stepping forward and backward, and turning its view direction around. Light direction also changes during the motion. The distribution of θ is influenced by light and view changing and irrelative to the scene used. From this figure we can see that θ located in the range of 25 degrees to 30 degrees at a large proportion. This is because characters in a movie or game usually take a great proportion of time at stepping forward or backward. Actually, θ is never greater than 40 degrees in all experiments.

Figure 5 shows the average partition number influenced by ζ in a same wandering path. We can learn from this figure that the average partitions needed for shadow rendering decreases with the increment of the wasting factor. The value of ζ is set scene-independently by considering the tradeoff between image quality and rendering efficiency.

To demonstrate the advantage of our LSPSMs method in rendering efficiency, we compare it with latest contributions, including LiSCSMs [7], SDSMs [6] and OPCSMs in CryENGINE [8]. In our LSPSMs, uv resolution difference tolerance factor is set as $\sigma = 0.5$. The maximum lengthways partition number is limited as $h_{max} \leq 4$ and the transverse partition number is limited as $N_w \leq 2$. The wasting factor ζ is set as $\zeta = 0.25$, a value which is proved helpful for both high quality and efficiency, after repeated experiments. For all the competent algorithms, the splitting number in Z-direction is 8. Under such parameter settings, LSPSMs will consume same(8) rendering passes at most with LiSCSMs and SDSMs for the worst case. Four scenes are used in our comparisons, including a Tree scenario, a Power Plant, a Sponza and a Geometries scenario. Table 1 demonstrates the comparison result in the average partition number and average frame time (in millisecond).

Among all the competent methods, our LSPSMs and CryENGINE have the ability to eliminate the overlapping problem. But as Table 1 shows, CryEngine consumes nearly 3 times rendering passes to our LSPSMs in average, making its efficiency much lower. Only in the indoor scene of Sponza, SDSMs shows best performance due to its ability to create the tightest light space bounding to cull the occluded objects. In all the other cases, our LSPSMs get better performance than the competent algorithms because less rendering passes are consumed in average.

4.2 Quality

LSPSMs has the ability to produce shadow in nearly constant "texel-per-pixel" ratio. Suppose the resolution in the far transverse side of each shadow map is same with screen, from Eq. (1) we know that the maximum screen error in this shadow maps is less than ζ $(\frac{2H_i \tan\theta}{W_i} \leq \zeta)$. The efficient texture space utilization and the high sample density make the shadow quality rendered by LSPSMs in high precision.

Table 1 Comparison of average partition number (N) and average frame time (T, in milliseconds) in different scenes ($\zeta = 0.25$).

Scene	LSPSMs		LiSCSMs		SDSMs		OPCSMs	
	N	$T(ms)$	N	$T(ms)$	N	$T(ms)$	N	$T(ms)$
Tree	4.32	5.19	8	9.26	8	9.14	17	10.02
PowerPlant	5.61	6.07	8	11.53	8	12.37	17	15.53
Sponza	5.13	6.26	8	10.95	8	5.78	17	15.14
Geometries	6.15	7.39	8	9.54	8	8.03	17	11.25

Fig. 6 Quality of shadow edges are finer in the images rendered with transverse partitioning (left) than images rendered without transverse partitioning.

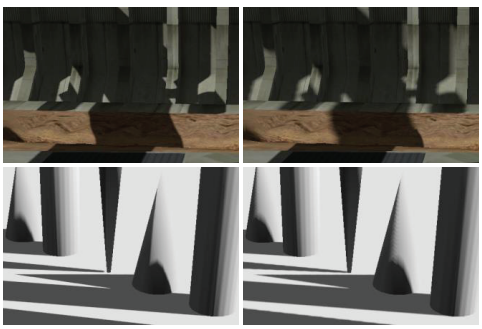

Fig. 7 Compared to LiSCSMs (right), LSPSMs (left) can render more subtle details. Notice the comparison of shadow edges. Top row: power plant scene. Bottom row: geometries scene.

With the help of transverse partitioning, subtle details can be rendered in high precision even in the case of the viewer being very close the shadows. As Fig. 6 shows, better quality can be seen at the shadow edges from the image produced with transverse partitioning.

Compared to LiSCSMs, our LSPSMs can create shadow maps with higher sample density. The improvement of shadow quality can be demonstrated by comparing it with LiSCSMs, as Fig. 7 shows. More improvements can be see in our supplemental images and video.

5. Conclusions and Future Works

We propose Light Space Partitioned Shadow Maps algorithm (LSPSMs) in this letter for high quality shadow rendering. LSPSMs split the projective region into partitions in light space, which represent viewports of light frusta. The partitioning operation is performed under the limitation of wasted space, rather than by calculating Z-partitioning points like traditional methods. The problem of space overlapping is eliminated for high sample density. Partitions are crated in dynamic number corresponding to the light and view directions for efficient shadow rendering. Experimental results show that LSPSMs raise the rendering efficiency while keeping high image quality.

In the future, we would be interested in exploring the combination of LSPSMs with warping techniques for finer image quality. Algorithms that attempt to cast shadows on a spherical and arbitrary surface are of particular interest.

Acknowledgements

This research work was supported by the Jiangsu Province Science Foundation for Youths of China (Grant No. BK20150722). Thanks to NVIDIA to provide the Tree scene and thanks to Microsoft to provide the Power Plant scene, Sponza scene, and Geometries scene in their SDKs. We would like to thank all the reviewers of this letter for their valuable and constructive comments.

References

[1] L. Williams, "Casting curved shadows on curved surfaces," ACM Siggraph Computer Graphics, pp.270–274, ACM, 1978.
[2] E. Eisemann, U. Assarsson, M. Schwarz, M. Valient, and M. Wimmer, "Efficient real-time shadows," ACM SIGGRAPH 2013 Courses, SIGGRAPH '13, New York, NY, USA, pp.18:1–18:54, ACM, 2013.
[3] W. Engel, Cascaded shadow maps, pp.197–206, Charles River Media, 2006.
[4] F. Zhang, H. Sun, L. Xu, and L.K. Lun, "Parallel-split shadow maps for large-scale virtual environments," Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications, VRCIA '06, New York, NY, USA, pp.311–318, ACM, 2006.
[5] D.B. Lloyd, D. Tuft, S.e. Yoon, and D. Manocha, "Warping and partitioning for low error shadow maps," Proc. 17th Eurographics Conference on Rendering Techniques, EGSR '06, Aire-la-Ville, Switzerland, Switzerland, pp.215–226, Eurographics Association, 2006.
[6] A. Lauritzen, M. Salvi, and A. Lefohn, "Sample distribution shadow maps," Symposium on Interactive 3D Graphics and Games, I3D '11, New York, NY, USA, pp.97–102, ACM, 2011.
[7] X.-H. Liang, S. Ma, L.-X. Cen, and Z. Yu, "Light space cascaded shadow maps algorithm for real time rendering," Journal of Computer Science and Technology, vol.26, no.1, pp.176–186, 2011.
[8] N. Kasyan, "Playing with real-time shadows." http://www.crytek.com/cryengine/presentations/playing-with-real-time-shadows, 2013.