A Shadow Culling Algorithm for Interactive Ray Tracing

Jae-ho Nah† Kyung-ho Lee‡ Woo-chan Park⋆ Tack-don Han†

†Yonsei Unveriisty ‡Institut Pasteur Korea ⋆Sejong University

Abstract

We present a novel shadow culling algorithm in ray tracing. For interactive ray tracing, our approach exploits frame-to-frame coherence instead of preprocessing of building shadow data. In this algorithm, shadow results are stored to each primitive and used in the next frames. We present a novel occlusion testing method to guarantee exact shadows. In experiments with BART scenes, our algorithm shows 7-19 percent reduction in cost of traversal and 9-24 percent reduction in cost of intersection test.

Keywords: Ray tracing, real-time rendering, shadow algorithm,
Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Ray tracing

1 Introduction

According to Whitted-style ray tracing algorithm, the number of shadow rays is relative to the number of light sources. For example, if there are nine point light sources, the rate of shadow rays is 90 percent because a non-shadow ray (primary ray, reflection ray, and refraction ray) generates nine shadow rays. Thus, efficient shadow processing is essential for ray tracing.

To reduce the shadow calculation, there are several preprocessing approaches. The light buffer [1] stores object list to grid light structure. Voxel occlusion test [2] stores shadow results to grid acceleration structure. Local illumination environments (LIEs) [3] caches geometric and direct illumination to octree acceleration structure. Lightcuts [4] is binary light tree of clusters to cut unnecessary lighting. These preprocessing methods are only useful in static scenes because they need to update the shadow results from scratch every frame in dynamic scenes. Haines et al. also presented shadow caching which exploits object coherence in [1]. This algorithm can be applied irrespective of preprocessing, but it is penalized in incoherent shadow rays and non-occluded shadow rays. [5]

In this paper, we propose a novel shadow culling algorithm which utilizes frame-to-frame coherence for interactive ray tracing. For the utilization, shadow results are accumulated on each primitive to reduce redundant shadow tracing in the next frame. If shadows are changed because of dynamic objects, the shadow results are partially updated in the only related primitives.

Updating shadow results is valid if and only if the primitive is not occluded on the view point because the shadow results on occluded area don't be calculated. To detect the occlusion, we also present a novel occlusion testing technique which needs simple comparison between hit primitives on adjacent pixels.

Our algorithm offers the following benefits. First, our algorithm efficiently reduces the cost of shadow rendering. Second, our algorithm doesn't need preprocessing. Third, our algorithm is easy to implement and it has small overhead.

E-mail : jhnah@msl.yonsei.ac.kr

2 Proposed Shadow Culling Algorithm

2.1 Algorithm Flow

Figure 1 is the overall processing flow of our algorithm. It is similar to traditional ray tracing algorithm except some modifications. First, the occlusion testing stage is added to carries out occlusion testing to decide occlusion modes on partially occluded primitives. Next, the shadow testing stage is modified to reduce redundant calculation. After the shadow testing, the results set the shadow mode on the hit primitive. Finally, the occlusion mode setting stage decides occlusion modes on fully rendered primitives.

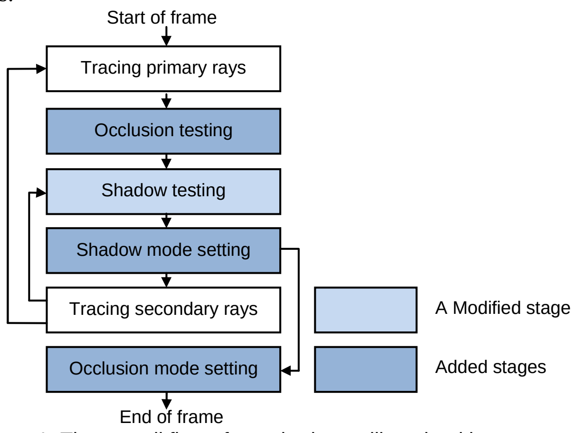

Figure 1: The overall flow of our shadow culling algorithm

2.2 Required Additional Information

Each primitive needs the additional information to reuse shadow results: an occlusion mode (2bit), a shadow mode (2bit*light sources), the number of current rendered pixel (1 integer), and the highest number of rendered pixel (1 integer). Figure 2 represents the occlusion mode and the shadow mode. The occlusion mode and the shadow mode need only a few bits, so they can be merged to a 32bit integer in less than ten lights.

Occlusion mode (value)	
	INIT(00)
⊿	PARTIALLY_ OCCLUDED(01)
△	FULLY_ RENDERED(10)
↻△	DYNAMIC(11)

Shadow mode (value)	
	INIT(00)
▲	FULL_SHADOW (01)
△	NON_SHADOW (10)
◭	PARTIAL_ SHADOW(11)

Figure 2: (Left) four occlusion modes (Right) four shadow modes

2.3 Occlusion Testing

In this section, we propose a novel occlusion testing algorithm. The occlusion testing method is simple. It compares primitives between the current pixel and two reference pixels. If the primitives are different, an intersection test is executed between the current primary ray and the primitive on the reference pixel. If the hit result is true, the primitive on the reference pixel is partially occluded by the primitive on the current pixel. If the current pixel is located at the edge of screen, the primitive on the current pixel has always the PARTIALLY_OCCLUDED mode because the primitive is clipped by the screen.

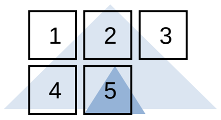

Figure 3: An example of the occlusion test – if the current rendering pixel is 5, two occlusion testing are executed about the pixel 4(the left pixel of 5) and the pixel 2(the upper pixel of 5). After the testing, the renderer detects that the faint blue triangle is occluded by the dark blue triangle.

2.4 Shadow Testing

The shadow testing stage detects shadows on the current shadow ray, and also decides whether shadow culling can be executed or not. If a primitive has FULLY_RENDERED mode and FULL_SHADOW mode, the shadow tracing is skipped. If the shadow mode is NON_SHADOW, other dynamic objects could make a shadow. In this case, the current shadow ray should trace dynamic objects. If the node hasn't dynamic objects, traversal will be stopped. Because of this feature, the acceleration structure construction from scene graph [6] is very suitable in our algorithm. In other words, our algorithm has best efficiency when split planes are object boundaries instead of primitive boundaries.

2.5 Shadow Mode Setting

As we mentioned in Section 2.2, the size of shadow mode is two bit. The first bit means shadow and the second bit means non-shadow. Therefore, shadow mode setting can be executed with a simple OR operation. If a primitive has shadow partially, the shadow mode of the primitive become 11(PARTIAL_SHADOW) after an OR operation between 01(FULL_SHADOW) and 10(NON_SHADOW).

2.6 Occlusion Mode Setting

This stage set occlusion modes of rendered primitives before the end of frame. If the occlusion mode of a primitive is INIT at that time, the mode is changed to FULLY_RENDERED. It means that the primitive doesn't occluded by any other primitives. On the contrary, if the occlusion mode of a primitive is FULLY_RENDERED and the number of current rendered pixel is more than the quadruple of the highest number of rendered pixel, the occlusion mode is changed to INIT and the highest number of rendered pixel is updated. This processing prevents shadow artefacts on zoom-in as figure 4.

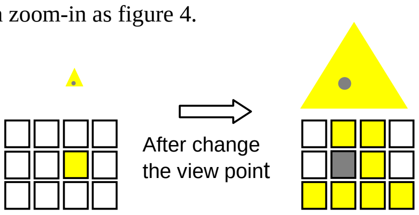

Figure 4: Upper line images are original primitives, and lower line images are projected images on pixels. After change the view point, invisible shadow on the primitive could be visible. Our algorithm solves this consistency problem on the occlusion mode setting.

3 Experiment Results

We implemented a Whitted-style ray tracer to verify our algorithm. The acceleration structure is kd-tree which has a split criterion between dynamic and static objects using scene graph.

We selected the BART scenes (Kitchen and Robot) for benchmark. The Kitchen scene is composed of 100K static triangles and 10K dynamic triangles. The Robot scene is composed of 10K static triangles and 60K dynamic triangles. The benchmark was executed on 512×512 resolution and maximum recursion depth was 10. The experiment metric is the number of traversal and intersection test.

Table 1 represents cost reduction results. Our algorithm is more efficient in the Kitchen scene than the Robot scene. The reason is that the former is static object-centred but the latter is dynamic object-centred. Table 1 also includes the overhead of our algorithm. Our algorithm needs three additional stages as figure 1. The shadow mode setting and the occlusion mode setting require only simple calculation but occlusion test requires additional ray-primitive intersection tests. Thus, we measured this occlusion test overhead. According to experiment results, the overhead is very low because additional tests only occur on the primitive boundaries when primary rays are traced.

Scene	Cost reduction ratio of shadow ray traversal		Cost reduction ratio of shadow ray/primitive intersection test		Additional intersection tests for occlusion testing
	Avg.	Max.	Avg.	Max.	
BART Kitchen	19.8%	59.5% (20th frame)	24.5%	59.8% (19th frame)	0.173 per pixel
BART Robot	7.4%	19.5% (31th frame)	9.5%	29.6% (41th frame)	0.149 per pixel

Table 1: Experiment results in BART kitchen and robot scenes

4 Conclusions and Future Works

We presented a novel shadow culling algorithm for interactive ray tracing. This algorithm accumulates shadow results and reuses the results to exploit frame to frame coherence. Thus, it doesn't need preprocessing in contrast priori works. Our algorithm has some overhead for occlusion testing, but it is quite low.

Many other acceleration algorithms can be mixed with our algorithm, and it would make synergy effects. Shadow caching would effectively reduce the shadow calculation of partially occluded primitives, and triangle subdivision would reduce the ratio of primitives which have partial shadows. These hybrid approaches will be examined in a future publication.

References

[1] E.A.Haines and D.P.Greenberg, The light buffer: A shadow testing accelerator. *IEEE Computer Graphics and Applications*, vol. 6, no. 9, pages 6-16, Sep 1986.

[2] A.Woo and J.Amanatides, Voxel occlusion testing: A shadow determination accelerator for ray tracing. *Proceedings of Graphics Interface '90*, pages 213-226, May 1990.

[3] S.Fernandez, K.Bala, and D.P.Greenberg, Local illumination environments for direct lighting acceleration. *Proeedings of 13th Eurographics workshop on Rendering*, pages 7-14, June 2002.

[4] B.Walter, S.Fernandez, A.Arbree, K.Bala, M.Donikian, and D.P.Greenberg, Lightcuts: A scalable approach to illumination. *ACM Transaction on Graphics*, vol.24, no.3, pages 1098-1107, Aug 2005.

[5] I.Wald, Realtime Ray Tracing and Interactive Global Illumination, Ph.d thesis, Sarrland University, 2004.

[6] W.Hunt, W.R.Mark, and D.S.Fussel, Fast and lazy build of acceleration structures from scene hierarchies. *Proceedings of IEEE/ EG symposium on interactive ray tracing*, pages 47-54. Sep 2007.