

Soft Shadow Maps for Area Light by Area Approximation

Zhengming Ying, Min Tang, Jinxiang Dong

State Key Laboratory of CAD&CG, AI Institute, Zhejiang University, Hangzhou 310027

Abstract

Shadow mapping has been a popular way of generating shadows in real-time applications, but it is still incapable of generating realistic real time soft shadows for area light. There is an algorithm for generating soft shadows for linear light with few samples, but the algorithm is not suitable for area light. In this article, the author presents a modified method to the shadow mapping algorithm for linear light. And the new method can produce convincing soft shadows for area light in real time applications.

1. Introduction

Shadows play an important role of indicating shapes, relative positions and other characteristics of objects and lights. The shadow algorithms fall into several categories according to Woo's survey [6].

Shadow mapping [3,4,5] is one of the shadow algorithms that are widely used in today's applications. As it for soft shadows, conventional method is sampling the area light and averaging the results, but the method is too expensive for real time applications.

An algorithm of soft shadows for linear lights [1] overcomes the problem of the sampling-and-averaging problem by interpolating the shadow intensity between ends of a linear light. Only two samples on a linear light can provide continuous realistic soft shadows. However, the algorithm has an artifact for area light: the transition between fully shadowed and fully illuminated area is linear, which is not a good approximation of soft shadows for area lights.

Our algorithm is a modification to the linear light algorithm, and can produce quadric shadow transition for area light.

2. Soft shadow map for area light

Our algorithm is an extension to the Heidrich's shadow map algorithm [1], and our area approximation method overcomes the shortcomings of the algorithm. It can be performed in several steps:

First, we sample the area light along the edges. The sampling points must be on the same plane and form a convex polygon for area approximation, or we have to divide the area light into smaller planar ones. Suppose the samples are named from P_0 to P_n .

Then, we can regard the points as spotlight, take a view from the points, and store the depth map individually. The above two approaches is pre-processing steps.

When we render the scene, for each pixel on screen P_e , we can use depth map algorithm to find if P_e is visible from light P_i . The Boolean value is stored in $V_p(P_i)$. And we can use a slightly modified method from Heidrich's to find the visibility value for each segment. This float value is stored in $V_s(P_i)$. Then approximate the area of the area light can be seen from P_e , that is S_e . Thus, S_e/S_a (total area of area light) is the intensity of shadow, where 0 indicates total in shadow, and 1 indicates totally not in shadow.

Then the shadow value can modulate the light to generate soft shadows. From above, we can see that our algorithm only add to the rendering time when calculating the area. Next we propose an algorithm that approximates the area in a few multiplications and adds.

In a typical circumstance such as in Figure 1, by iterating through the points of the polygon, we can calculate the hatched area S_k . We can use S_k or $1-S_k$ as S_e .

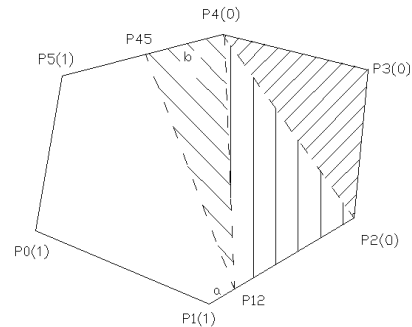


Figure 1

The number in parenthesis after the vertex is the value of $V_p(P_i)$. $V_s(P_1)=a$, $V_s(P_4)=b$, and other $V_s(P_i)$ values

are obvious. We want to calculate the area of hatched polygons, and have the following equation:

$$S_h = S(P_2P_3P_4) + S(P_{12}P_2P_4) + S(P_4P_{45}P_{12}) \\ = S(P_2P_3P_4) + S(P_1P_2P_4) * (1-a) + b * (S(P_1P_4P_5) + (S(P_2P_4P_5) - S(P_1P_4P_5)) * a)$$

We can compute $S(P_xP_yP_z)$ in the second equation beforehand. Such a fast approximation method can reduce the calculation of area to several multiplies and additions, causing a slight increase in rendering time.

If we increase the samples on an area light, the pre-processing time increases by the number of samples. From the above equation for area approximation, we can see the rendering time will not increase.

3. Our implementation

We use OpenGL to implement the algorithm. Except one pass for ambient light, no matter how many points the area light has been sampled, it can be rendered in one pass, but different light must be rendered in different pass.

Below are four samples from the demo program. The depth map and visibility channel resolution is 512x512. The area light is a triangle. Pic. 2(a) uses three spotlights. Pic. 2(b) uses two vertexes of the triangle and forms a linear light. Pic. 2(c) uses the edges of the triangles as three linear lights. Pic. 2(d) uses the vertices of the triangle as sampling points of the area light. We can see the falloff of shadow intensity of the lower-right picture is more realistic than lower left picture.

Both our method and linear light method need to transform the point in eye coordinate into light coordinate,

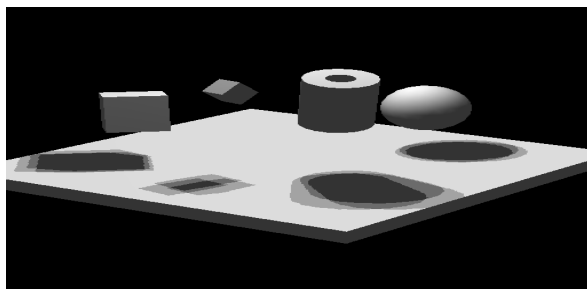
and our area approximation method only takes a little time compared to coordinate transformation. So our algorithm keeps the rendering time within real time applications.

4. Conclusion and future work

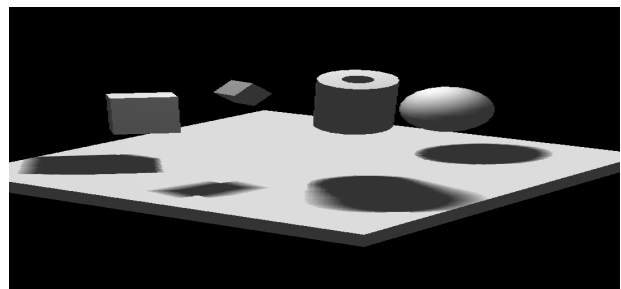
In this article, an approximation of soft shadow maps has been proposed. The slightly increased computation will not affect its real-time application. And only few samples can have convincing soft shadow of the scene. These features made it suitable in real-time soft shadow generation.

5. Reference

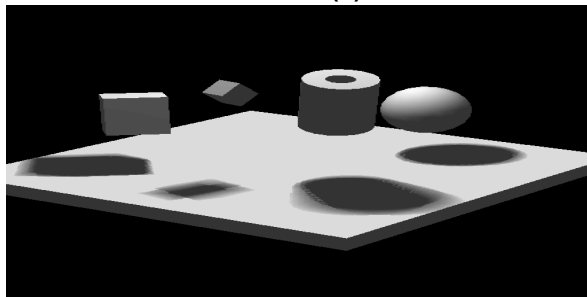
- [1] W. Heidrich, S. Brabec, H. Seidel, Soft Shadow Maps for Linear Lights, *Proc. Eurographics Workshop on Rendering'00*, Springer, 2000.
- [2] M. D. McCool. Shadow Volume Reconstruction from Depth Maps, *ACM Transactions on Graphics*, 19(1), Jan 2000
- [3] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. *Computer Graphics (SIGGRAPH '87 Proceedings)*, July 1987.
- [4] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haerberli. Fast shadow and lighting effects using texture mapping. *Computer Graphics (SIGGRAPH '92 Proceedings)*, July 1992.
- [5] L. Williams. Casting curved shadows on curved surfaces. *Computer Graphics (SIGGRAPH'78 Proceedings)*, August 1978.
- [6] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE CG&A*, November 1990.



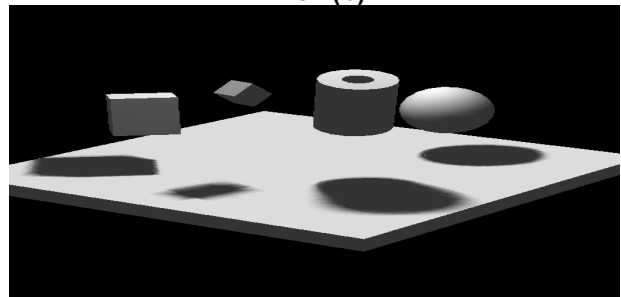
Pic 2(a)



Pic 2(b)



Pic 2(c)



Pic 2(d)

Figure 2