An Adaptive Subdivision by Sliding Boundary Surfaces for Fast Ray Tracing

Keiji NEMOTO and Takao OMACHI

C&C Systems Research Laboratories, NEC Corporation

4-1-1 Miyazaki, Miyamae-ku, Kawasaki, Kanagawa, 213 Japan

(044) 855-1111

ABSTRACT

This paper presents an adaptive subdivision algorithm for fast ray tracing implemented on parallel architecture using a three dimensional computer array. The object space is divided into several subregions and boundary surfaces for the subregions are adaptively slid to redistribute loads of the computers uniformly. Since the shape of the subregions is preserved as orthogonal parallelepiped the redistribution overhead can be kept small. The algorithm is quite simple but can avoid load concentration to a particular computer.

Simulation results reveal that the adaptive space subdivision algorithm by sliding boundary surfaces reduces the computing time to 3/4-1/5 as much as that for the conventional space subdivision algorithm with no redistribution, which reduces the computing time almost proportionally to the number of the computers.

KEYWORDS: sliding, adaptive, parallel, ray tracing, subdivision, boundary.

Introduction

Among general image synthesis methods available today, ray tracing[1] is probably the most realistic technique, because it models a wide range of natural phenomena. However, it requires a large amount of computing time. The calculation for ray-object intersections requires 75-95 percent of the total computing time[1].

Various approaches have been attempted toward speeding up of ray tracing. Previous research reports are categorized as follows:

(1) Multicomputer system by image subdivision[2]: An image to be generated is divided into several subimages and each of the computers generates one or more subimages independently.

(2) Vectorization[3]: Since the ray-object intersection calculations belonging to the different pixels and the intensities of the different pixels are calculated completely independently, the calculations can be vectorized.

(3) Space subdivision[4,5,6]: The three dimensional space of a scene to be rendered is divided into subregions. The rays which are cast into each subregion are tested for intersection with the objects contained within the subregion. Data on rays that exit a subregion are passed to the appropriate neighbor.

The first two ways do not reduce the number of ray-object intersection calculations, but speed up the intersection process itself, by parallel processing and specialized hardware.

On the other hand, the space subdivision method can reduce the number of calculations, because it tests rays for intersection only with the objects contained within the subregions that rays pass through, instead of all objects in the entire scene.

Recent work has applied a parallel architecture to this space subdivision algorithm[4]. This architecture uses a three dimensional computer array, each computer of which is assigned to one or more subregions. The shapes of the subregions are "general cubes", which are general hexahedron, and the shapes are adaptively controlled to realize a roughly uniform load distribution. This algorithm has the following problems:

1) Load is transferred among the subregions by moving corners of a general cube indicating the subregion. The moving operation to distribute the load is quite difficult because moving one corner affects the loads of the eight subregions holding it in common. The problem is how to select the corner to be moved and how to determine the direction and the length to move the corner in a three dimensional space in order to distribute the loads of the eight subregions at once.

2) When rays exit the subregion, the neighboring subregion that rays are passed to is determined by boundary-intersection calculation. However, boundary-intersection testing for general cubes is a significant overhead.

3) When a corner of the subregion is moved, an