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

appropriate part of the object descriptions contained within the subregion are determined and pertinent data is passed to the neighboring subregions. This operation is as expensive as boundary-intersection testing.

A moving corner method would greatly affect the performance of this algorithm. However, the problem how to move the corner to distribute the load cannot be easily solved because of the difficulties mentioned above.

This paper presents a new approach to solve the problems above. The shapes of the subregions are limited to orthogonal parallelepipeds, and load is transferred by sliding boundary surfaces of the subregions.

The following sections will discuss a simple subdivision algorithm for parallel architecture and give simulation results.

New Space Subdivision Algorithm

The essential algorithm characteristics are:

1) The three dimensional space of a scene to be rendered is divided into several subregions by planes perpendicular to a coordinate axis(Fig.1). Positions x_i, y_i, and z_i of the dividing planes have integer coordinate values. Thus, each subregion is an orthogonal parallelepiped which consists of several unit cubes. A unit cube is a cube whose size is 1 and whose edges are parallel to each coordinate axis.

2) Each computer of three dimensional computer array is assigned to one subregion and maintains only the object descriptions contained within that subregion.

3) Each computer has 6 connections to neighboring computers in order to pass messages (Fig.2), which consist of information about rays and redistribution. Each computer also has a direct connection to a host computer and to a frame buffer. Messages regarding object descriptions are directly sent from a host computer to all computers as broadcast messages. Each computer determines which object is contained within its own subregion and preserves only its description.

4) Initial rays from the eye point are created by all computers in parallel. An image to be generated is divided into subimages and each computer is assigned to one of the subimages. Each computer creates the initial rays which pass through its own subimage. Then each initial ray is transferred to the appropriate subregion where the initial ray starts. After each of the rays has reached to the appropriate subregion, it is tested for intersection with those objects within the subregion.

5) Rays that exit the subregion are passed to neighboring subregions via connection between computers. The three dimensional digital line[6] is generated for efficient tracing of rays. The three dimensional digital line is an array of unit cubes pierced by the ray (Fig.3).

To determine the the array of unit cubes, two DDAs (Digital Differential Analyzers)[6] generate two digital lines synchronously which are the projections of the three dimensional digital line to two coordinate planes. The error values of two DDAs are compared to decide the correct direction of the three dimensional digital line (Fig.4).

Since the subregion consists of several unit cubes and the three dimensional digital line is generated only by addition of integer values, rays can rapidly traverse the subregion and also enter the appropriate neighboring subregion by using the three dimensional digital line.

6) For redistribution, the boundary surface between two subregions is slid by one unit and a

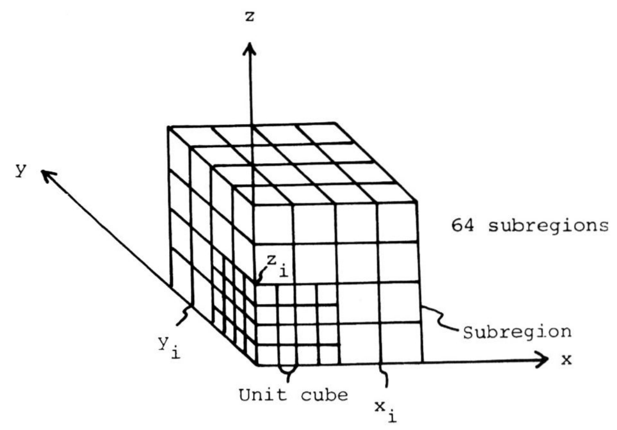

Figure 1. Three dimensional space division

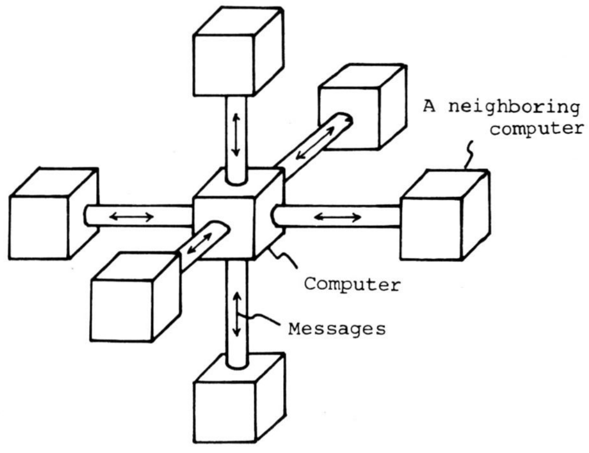

Figure 2. 6 connections from a computer

part of the load for one subregion is transferred to the other subregion. The following section gives more details about sliding boundary surfaces.

Sliding Boundary Surfaces

The adaptive subdivision by sliding boundary surfaces is as follows:

1) At the beginning, one of three coordinates axes is set as a driving axis (e.g. x axis in Fig.5). Boundary surfaces for the subregion perpendicular to the driving axis are slid by one unit along the driving axis to transfer the load from one subregion to a neighboring subregion.

The subregion load is related to the number of the objects contained within the subregion. Therefore, the axis along which the numbers of the objects in the subregions are most varied is set as a driving axis.

2) Each computer counts the running time while the computer actually processes the rays. Each computer also counts the waiting time while the computer has no ray to be processed and is waiting the rays passed from the neighboring computers. The ratio (running time) / (waiting time) is defined as a parameter to indicate the load for the subregion.

3) For redistribution, loads for two subregions on both sides of the boundary surface are compared by the computers assigned to these two subregions at intervals of the given time . If the load for one subregion is lower than that for the other subregion and the lower load is under the given threshold value, the boundary surface is slid

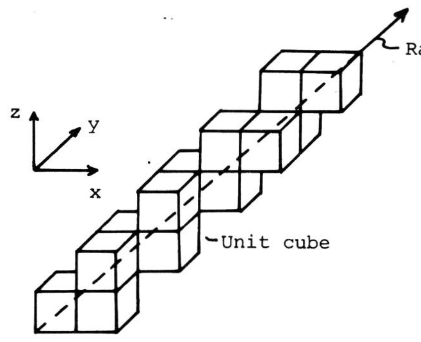

Figure 3. Three dimensional digital line

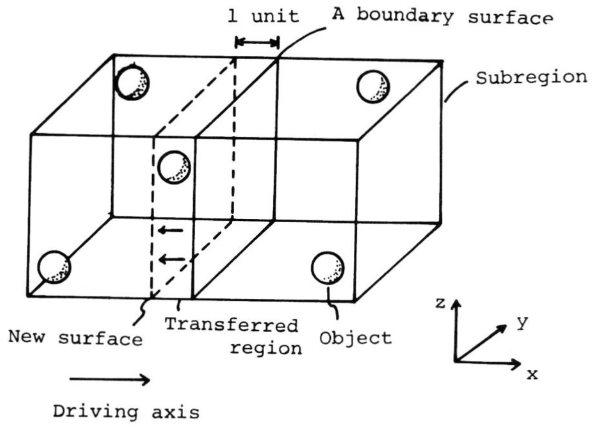

Figure 5. Sliding a boundary surface

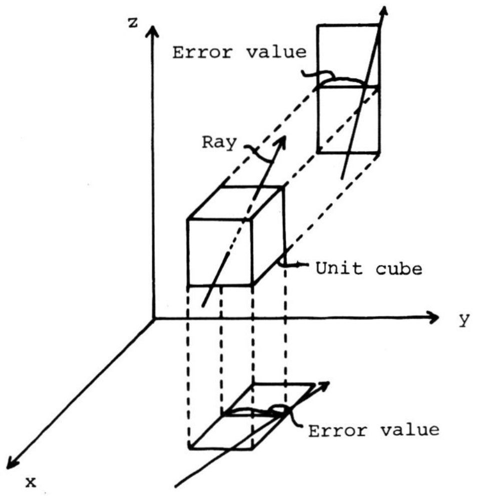

Figure 4. Error values of two DDAs

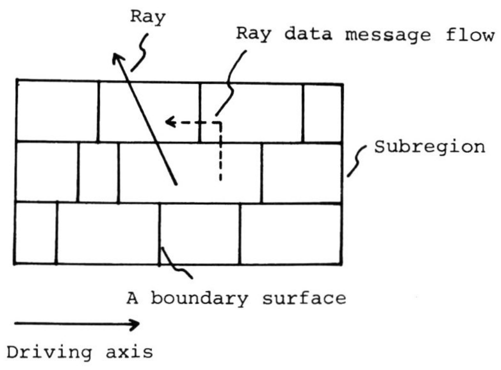

Figure 6. Boundary surfaces discrepancy (Two dimensional view)

by one unit along the driving axis from the lower load subregion to the higher load subregion. By this operation, a part of the subregion with the higher load (called the transferred region) is cut away and added to the lower one. Simultaneously, the object descriptions contained within the transferred region are transferred. Thus, the load is simply and efficiently transferred. These operations are locally executed by the computers assigned to these two subregions.

4) As each boundary surface is slid for the redistribution, some discrepancy in the boundary surface occurs (Fig.6). However, since the connections between computers are fixed, there could be a case wherein a computer has no direct connection to another computer assigned to the neighboring subregion. In this case, data concerning the rays that exit the subregion cannot be directly passed to the appropriate computer assigned to the appropriate subregion, so that data on rays are passed to the direct connected computer which is assigned to the subregion located on the same driving axis with the appropriate subregion (shown by a dotted line in Fig.6). After that, data on rays are passed along the driving axis where they can finally reach the appropriate computer.

Essential characteristics of the method are as follows:

1) For redistribution, only the loads for two subregions on both sides of the boundary surface perpendicular to the driving axis are compared. Thus, the redistribution between the subregions is easily determined.

2) The shape of the subregions is preserved as an orthogonal parallelepiped. Since boundary surfaces are rectangular and perpendicular to coordinate axes, boundary-intersection testing is a small overhead.

3) Only the boundary surfaces perpendicular to the fixed driving axis are slid along the axis by one unit and the object descriptions contained within that transferred region are transferred. So the redistribution does not cause a significant overhead.

4) Since the given threshold value stops the sliding between the highly loaded subregions, a thrashing whereby the object descriptions are repeatedly moved between the highly loaded computers is avoided. A thrashing between the lightly loaded subregions does not matter to the total performance.

5) Because of the simplicity of this method, it can be easily implemented on a three dimensional computer array.

Results

The proposed adaptive subdivision algorithm by sliding boundary surfaces is simulated to evaluate the redistribution effect. The simulation results for redistribution are compared with that for no redistribution when the load is concentrated to some particular computers.

Beforehand, the effect of the conventional space subdivision algorithm without redistribution is evaluated by simulation.

A. Simulation Methods

The algorithms have been written in a C program and tested on a Vax-11/780 under the Unix operating system. A parallel process simulator[7] has been created to evaluate the algorithms. The simulator virtually causes the computers to run in parallel and counts the running time and the waiting time of each computer to calculate the load. The simulator also counts the computing time for generating an image by parallel architecture.

Objects are only spheres. These spheres are described by their center position, radius, color, and reflecting parameters and these parameters are generated as uniform random numbers. Therefore, the objects are located in a space at random.

B. Space Subdivision Effect

First, the effect of the conventional space subdivision algorithm is evaluated.

Figure 7 shows the computing time of the space subdivision algorithm without redistribution.

Result A shows the computing time when only a single computer is assigned to all subregions. The algorithm implemented on a single computer reduces the computing time on the order of $S^{2/3}$ (S = the number of the subregions).

Result B shows the computing time when each computer of three dimensional computer array is assigned to one subregion but initial rays are created by the computer whose subregion contains the eye point. The difference between results A and B means the parallel processing effect using the three dimensional computer array.

Result C shows the computing time when the initial rays are also created by all computers in parallel. This space subdivision method with the parallel initial ray creation reduces the computing time on the order of $S^{1.5}$. The difference between results B and C means the parallel creation effect of the initial rays. The parallel initial ray creation is effective when S is large.

Figure 8 shows the computing time when the number of objects is changed in the case of result C in Fig. 7. The computing time can be reduces on almost the same order even if the number of the objects is changed.

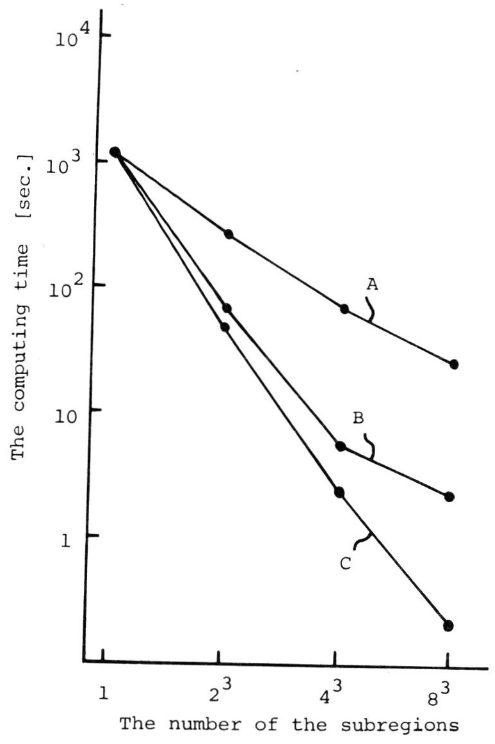

Figure 7. Space subdivision effect

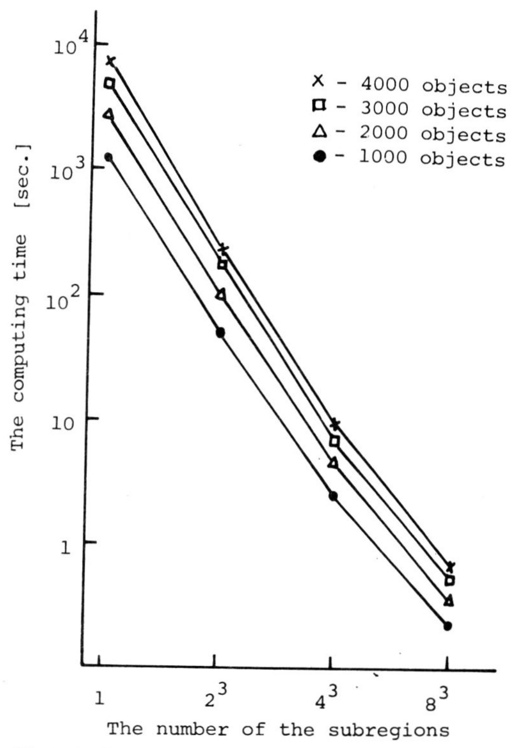

Figure 8. Space subdivision effect for several numbers of the objects

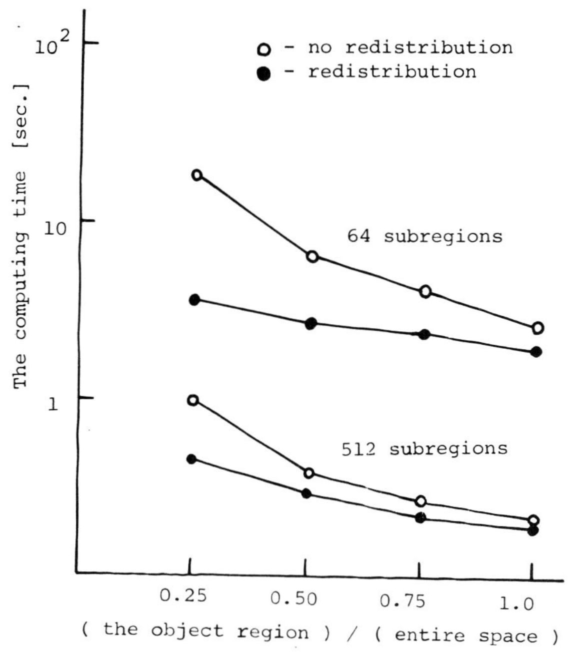

Figure 9. Redistribution effect for 1000 objects

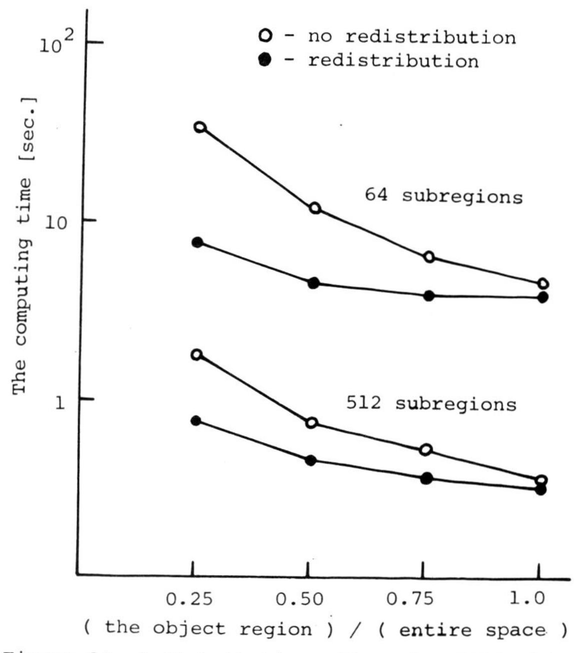

Figure 10. Redistribution effect for 2000 objects

C. Proposed Redistribution Effect

In order to examine the redistribution effect, the region where all objects are located is reduced from the entire space to quarter of the entire space. The smaller region the objects are located in, the more load is concentrated to some particular subregions.

Figures 9 and 10 show the comparisons between the computing time for redistribution by sliding boundary surfaces and that for no redistribution. The horizontal axes of Figs. 9 and 10 means the volume ratio of the region where all objects are located against the entire space.

When the objects are uniformly located in a space at random, the loads have been almost uniformly distributed initially so that the redistribution does not work so effectively. Even so, the redistribution can reduce the computing time to 3/4 as much as that for no redistribution.

The more concentrated the objects and the loads are to a part of the subregions, the greater the redistribution effect becomes. The effect becomes up to 1/5 when the objects are concentrated to the quarter of the entire space. these results mean that the redistribution by sliding boundary surfaces has an equivalent effect to distribute the concentrated objects to the entire space.

Moreover, Figs. 9 and 10 show almost same redistribution effect, so that not the number of the objects but the objects location in the space controls the redistribution effect.

Conclusions

This paper has presented a simple adaptive subdivision algorithm implemented on the parallel architecture using a three dimensional computer array. Boundary surfaces of 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. By using this algorithm, the computing time is reduced as much as 3/4-1/5 of that for no redistribution.

Acknowledgment

The authors would like to express appreciation for continuous encouragement from K.Niwa. They also acknowledge the significant contributions of S.Fukui and H.Kanazawa.

References

1. T.Whitted, "An Improved Illumination Models for Shaded Display," Comm. ACM, Vol.23, No.6, '80, pp.343-349.

2. H.Nishimura, H.Ohno, T.Kawata, I.Shirakawa, and K.Omuira, "LINKS-1: A Parallel Pipelined Multimicrocomputer System for Image Creation," Proc. of the 10th Symp. on Computer Architecture, SIGARCH, '83, pp.387-394.

3. D.Plunkett and M.Bailey, "The Vectorization of a Ray-Tracing Algorithm for Improved Execution Speed," IEEE CG&A, Aug. '85, pp.52-60.

4. M.Dippe and J.Swensen, "An Adaptive Subdivision Algorithm and Parallel Architecture for Realistic Image Synthesis," Computer Graphics, Vol.18, No.3, Jul. '84, pp.149-158.

5. A.Glassner, "Space Subdivision for Fast Ray Tracing," IEEE CG&A, Oct. '84, pp.15-22.

6. A.Fujimoto and K.Iwata, "Accelerated Ray Tracing," Proc. of CG Tokyo '85, T1-2.

7. C.Binding, "Cheap Concurrency in C," SIGPLAN Notices, Vol.20, No.9, Sep. '85.