利用 Hybrid-SDSM 結構加速光線追蹤塑型實體組合模型之研究
A Study of Accelerating Ray Tracing CSG Solids Using Hybrid-SDSM Structure

張孝賢 王宏達 王宗銘
Shou-Shang Chang Hong-Da Wang Chung-Ming Wang

國立中興大學資訊科學研究所
Institute of Computer Science National Chung Hsing University
Taichung, Taiwan 402, R.O.C.
(sschang, hdwang, cmwang)@cs.nchu.edu.tw

摘要

CSG 是計算機輔助設計中一種常用的方法。利用光線追蹤法可使 CSG 模型產生高品質的圖像，但光線追蹤法卻會耗費大量的計算時間。我們提出一個 Hybrid-SDSM 資料結構來加快圖像的產生速度。Hybrid-SDSM 係結合空間分割與層狀界限容積兩種技術來加速光線追蹤。實驗證明 Hybrid-SDSM 確實能改善光線追蹤的計算時間。

關鍵字:光線追蹤法, Hybrid-SDSM

Abstract

Constructive Solid Geometry (CSG) is one of the most popular techniques to create 3D solids in Computer-Aided Design and Manufacturing. Besides, the ray tracing CSG solids can generate very realistic images with the expense of long computation costs. In this paper we will propose a hybrid-SDSM data structure which combines bounding volume hierarchy and spatial subdivision to accelerate ray tracing computing. The experiments collected demonstrate that the ray tracing time has been reduced using the proposed hybrid-SDSM data structure.

Key Words:Ray tracing, Hybrid-SDSM

1. Introduction

Solid modeling systems [11, 13, 14, 15] is used to represent three-dimensional components and assmblies. They make engineers to efficiently design, analyze and manufacture products [13, 15]. One of the requirements in these systems is to visualize models that are represented by the Constructive Solid Geometry (CSG) scheme [11, 15]. Realistic images are preferable to other display methods because they can help the designers reason with the solid models, identify model's weakness, and communicate the resultant model specifications to the manufacturers.

Ray tracing has been a popular and powerful display method generating high quality shaded images suitable for visualization purpose. The main idea of ray tracing is to trace rays backward from its source to pixels in an image plane, considering all possible intersections with the objects in the scene. The intensity of each pixel is calculated by the shading models and the characteristics of the object surfaces. Unfortunately, ray tracing takes a lot of time in ray-object intersection computations. Whitted et al. [19] estimated that in a ray tracing algorithm most of the time (75% ~ 95%) is spent in computing ray-object intersections to produce an image. While many techniques being presented for fast ray tracing, the space subdivision and the bounding volume hierarchy are the most popular schemes.

The main idea of space subdivision is to divide the image space into some disjoint cells. Each cell contains the objects in the model that intersect the space occupied by the cell. When the ray pierces a given cell, it only checks intersections with the objects in that cell. If there is no object in that cell or there is no intersection with all objects in that cell, the ray moves to the next cell on its path and repeats the same procedure mentioned above. Usually space subdivision technique can be divided into two parts : non-uniform and uniform schemes. Non-uniform space subdivision scheme divides the space into a set of non-overlapped cells according to the object distribution density. Uniform space subdivision scheme is to use the same grid without considering the object's distribution density.

The other technique to speed up ray tracing is called bounding volume hierarchy. The central idea behind this technique is to use a simple volume containing a given object and permits a simpler ray intersection check than the original object. Spheres or boxes are usually adopted as the bounding volumes and then are created bounds hierarchically. If a ray does not hit the parent bounding volume, then its child bounding volumes and objects within them need not to be checked. If a ray does hit the parent bounding volume, then the objects contained inside will be checked for the intersection.

This paper describes our approach to investigating techniques for fast visualizing CSG solids using ray tracing. A hybrid-SDSM data structure is proposed where both the spatial subdivision and the bounding

volume hierarchy are adopted. This paper is organized as below : section 2 describes previous work. Section 3 outlines the hybrid-SDSM structure. Some fundamental results are presented in section 4 followed by some conclusions and future work in final section.

2. Previous Work

Constructive Solid Geometry (CSG) is a scheme for representing solids using a set of primitive solids and the regularized set operations union ($\cup^*$), intersection ($\cap^*$) and difference ($-^*$). A model is generally represented as a binary tree whose internal nodes are set operations and leaf nodes are primitive solids.

The ray tracing algorithm was first described by Appel [1] and later by Goldstein and Nagel [8]. The basic operation of ray tracing can be divided into two phases. First, a primary ray is cast from each pixel for intersections with objects in the modeling space to determine the first visible surface. Second, pixel intensities are determined on the first visible surface by a shaded illumination model.

Due to long computation for ray tracing, we now review some related works about fast ray tracing.

Glassner [6] has implemented a ray tracing system for ordinary 3D objects. An octree-like data structure is used to sort the modeling primitives. 3D objects are maintained as a leaf cell representations and are accessed via a hash table. For each ray cast, the cells it intersects are determined incrementally until either a real ray-object intersection is found, or the ray exits the universal domain. Glassner improved the performance by factors ranging from 4 to 30 depending on the scene. This proposed data structure is very important, though Glassner's system did not support CSG modeling scheme.

Wyvill et al. [20] have also devised an octree-like structure to generate ray traced images of CSG models. They produced a divided model using octal cell division. The system is of interest because the spatial subdivision is controlled by both the halfspace number and a predefined maximum level of subdivision.

In 1991, Stephen Cameron [3] proposed a new efficient bound called S-bound. The main idea is to construct an efficient and tight bound of objects. The method can divide into two rules : upward rule and downward rule. After applying these rules several times, an efficient and tight bound can be generated.

Glassner [7] presented a new bounding volume hierarchy for efficiently ray tracing the animated scenes but not for CSG solids. It combined elements of non-uniform space subdivision and bounding volume hierarchy techniques.

Recently, Chuang and Hwang [4] proposed hybrid method for fast ray tracing CSG modeling scheme. The method constructs bounding box of CSG objects first. Then they applied spatial subdivision to the space according to the bounding box. Results showed the ray tracing time has been reduced. However, the model they proposed is less complexity containing at most 313 primitives.

3. A Hybrid-SDSM structure

This section describes the proposed data structure, hybrid-SDSM, used to accelerate ray tracing computations.

3.1 Creating a hybrid-SDSM structure

3.1.1 Creating bounding boxes of primitives

Each primitive's bounding box has to be determined after finishing creating a global CSG tree for the scene. We use three extent slabs whose unit normals are (1,0,0), (0,1,0), (0,0,1), respectively. We can construct a bounding box by intersecting a primitive and these three slabs. In our bounding volume hierarchy (BVH) structure, we only need to record the d value in a plane's equation; namely, ax + by + cz = d, as shown in Figure 1.

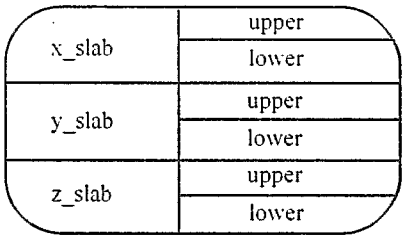

Figure 1. BVH Structure.

Recall that primitives in the modeling space may position in any location and orientation. The method of constructing bounding box for such primitive has to be determined. First, we create a bounding box along the primitive's orientation. Then we find out the eight corners' coordinates and get the extreme values of x, y, z. Finally these extreme values are the bounding box's d value.

After finding out each primitive's bounding box, we save these value in the leaf of CSG tree which preserves all informations about halfspaces. The structure is shown in Figure 2.

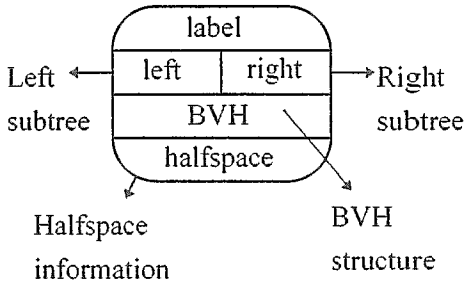

Figure 2. The CSG Structure.

3.1.2 Localized CSG tree Representation

The approach of non-uniform spatial subdivision of

modeling space is to adaptively subdivide modeling space and compile objects into the spatial hierarchy. For CSG modeling scheme, this means that it is necessary to develop the CSG tree a solid's local representation with respect to a spatial region such as cell. This process is referred to as the " tree pruning ", and a solid's local representation is called "localized CSG tree".

Consider an arbitrary cell V and a primitive P in 3D space. We identify such a cell as a **Full** cell if the cell V may be totally inside the primitive P. It may not associate with this primitive. We identify such a cell as an **Empty** cell. Finally, it may partially intersect with this primitive as being identified as a **Partial** cell. Figure 3 shows a CSG simplification lookup table. We simplify the global CSG tree into the localized CSG tree with respect to this cell V using the lookup table.

Operator	Left tree	Right tree	Simplified tree
Intesc. ∩*	Full	Full	Full
	Full	Partial	Partial (right tree)
	Full	Empty	Empty
	Partial	Full	Partial (left tree)
	Partial	Partial	Partial (left ∩* right)
	Partial	Empty	Empty
	Empty	Full	Empty

Operator	Left tree	Right tree	Simplified tree
Diff -*	Full	Full	Empty
	Full	Partial	Partial (left -* right)
	Full	Empty	Full
	Partial	Full	Empty
	Partial	Partial	Partial (left -* right)
	Partial	Empty	Partial (left tree)
	Empty	Full	Empty
	Empty	Partial	Empty

Operator	Left tree	Right tree	Simplified tree
Union ∪*	Full	Full	Full
	Full	Partial	Full
	Full	Empty	Full
	Partial	Full	Full
	Partial	Partial	Partial (left ∪* right)
	Partial	Empty	Partial (left tree)
	Empty	Full	Full
	Empty	Partial	Partial (right tree)
	Empty	Empty	Empty

Figure 3. The CSG Simplification Lookup Table.

3.1.3 Creating an SDSM

A universal cell containing the whole modeling space is first selected. A recursive decomposition of this cell is then employed together with the simplification of the global CSG tree with respect to each sub-cell generated. Each internal node of SDSM structure represents a **DIVIDED** cell and each leaf node represents a **TERMINAL** cell. A **DIVIDED** cell will maintain references to its eight sub-cells. A **TERMINAL** cell can be further characterized as a Full cell, an Empty cell or a Partial cell. Only a Partial cell maintains a reference to its localized CSG tree.

Figure 4 shows a " 2D " SDSM data structure. The SDSM cells corresponding to the adaptive spatial subdivision at each SDSM level, with the root cell being level zero, are expressed in clockwise starting from the upper-left-most cell. The DIVIDED, Partial, Full, Empty cells are abbreviated as D, P, F, E, respectively.

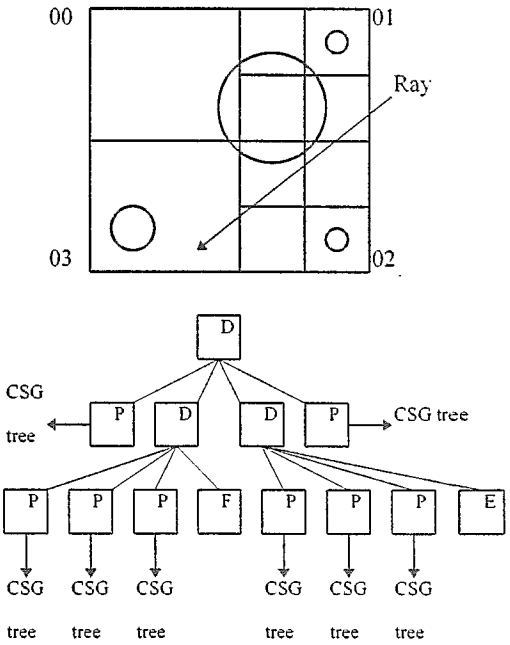

Figure 4. The Spatial Divided Solid Model.

3.1.4 Generating S-bounds

When an SDSM is being created, the next step is to generate an S-bound with respect to each localized CSG tree. We can divided this procedure into three phases. The first phase is the upward phase, the second one is the update phase, and finally the third one is the downward phase.

- *The upward phase* : This phase is to combine children's bounding box. It is done from the bottom to the top of the localized CSG tree. Let B(P) indicate the bound stored at a general node P in the localized CSG tree. Given a parent tree node P of the form P=C1☺C2. We can combine bounding box according to the parent operator as shown in Figure 5.
- *The update phase* : This phase is to update the

bounding box of the root of the localized CSG tree with respect to the cell boundary. Given the bound of the root for the localized CSG tree B(R) and the cell boundary B(C), we may update the bounding box of the root at this cell as B(R)∩B(C).

• *The downward phase* : In this phase we modify the children's bounding box from the top to the bottom using the root's bounding box. Given a child node C with a parent node P, we replace the bounding box at C by B(C)∩B(P).

☺ Operator	Left tree	Right tree	Combined bound
Union ∪*	∅	∅	∅
	∅	B(right tree)	B(right tree)
	B(left tree)	∅	B(left tree)
	B(left tree)	B(right tree)	Bl∪*Br
Intersc. ∩*	∅	∅	∅
	∅	B(right tree)	∅
	B(left tree)	∅	∅
	B(left tree)	B(right tree)	Bl∩*Br
Diff. -*	∅	∅	∅
	∅	B(right tree)	∅
	B(left tree)	∅	B(left tree)
	B(left tree)	B(right tree)	B(left tree)

Figure 5. The Lookup Table for Generating S-bound

3.2 Ray Tracing a Hybrid-SDSM

When a hybrid-SDSM data structure has been generated, the next step is to employ ray tracing algorithms on it to calculate the intensities of each pixel in the image plane. An outline of the ray tracing algorithm using a hybrid-SDSM structure, expressed as RayTraceSDSM(), is shown in Figure 6.

Given a hybrid-SDSM and a primary ray, the main ray trace process is executed for every pixel in the image. The procedure begins by calling RayTraceSDSM() function. Then it starts with RayNavigation() whose function is to traverse each cell along the path of the ray and proceed the ray-object intersection calculation.

```
RayTraceSDSM (SDSM, ray, nearest)
{
 hit = NOHIT;
 RayNavigation (ray->origin, SDSM->root, ray,
                hit, nearest);
 return(hit);
}

RayNavigation (point, cellptr, ray, hit, nearest) {
 if (PointOutofRange (point, cellptr) )
    return;
 currentnode = cellptr;
 if ( cellptr->kind == DIVIDED ) {
  childcellptr = LocateChildCell (point, cellptr);
  RayNavigation (point, childcellptr, ray, hit, nearest);
  if (hit == NOHIT )
    RayNavigation (point, currentnode, ray, hit, nearest); }
 else {
  FindRayCellRange (ray, cellptr->bounds, raycellrange);
  hit_bound = RayIntersectBVH (cellptr->tree->BVH, ray,
                               rayBVHrange);
  if ( hit_bound == HIT) {
   hit = RayCastCSG (cellptr->tree, ray, nearest,
                     rayBVHrange);
  }
  if (hit == NOHIT )
   GetNextPointOnRay (raycellrange, point, cellptr->bounds,
                      cellptr->level);
  else
   return;
 }
}
```

Figure 6. Ray Tracing Algorithm with A Hybrid-SDSM Structure.

When a ray is navigated within cells, we will first check if the cell is DIVIDED or Partial. If the cell is DIVIDED, then the ray is being navigated to the lower level of the hybrid-SDSM. Otherwise, a RayIntersectBVH() function is called to check the bounding box of the localized CSG tree. If a ray does hit the bounding box, then the ray-object intersection test is performed by calling the RayCastCSG() function. Otherwise, this ray is navigated to the next partial cell (if any) and the procedure just described will be continued.

4. Fundamental Results

The idea described above was implemented with the C programming language on UNIX-based workstation environments, and some results are collected here. These results are collected on SUN ULTRA-2 with 128M RAM.

Haines [9] proposed a set of models for evaluating the performance of systems using ray tracing for realistic image generation. Unfortunately, these models were not designed for CSG modeling scheme and most of them can not be easily represented by CSG.

We presented two test models here to show some results. The resolutions of images are 512×512 and two light sources only. Figure 7 lists parts informations of the test models. We perform the test five times for each test model and calculate the average timing values as shown in Figure 8 and 9.

Model	Resolution	No. of lights	Tree nodes	Halfspace
ball	512X512	2	1223	612
ngk	512X512	2	7895	3948

Model	Reflection	No. of test	Halfspace types
ball	No	5	sphere and cylinder
ngk	No	5	block, cone, cylinder and torus

Figure 7. Model Characteristics.

d:h	Bound	Preprocess time	Ray trace time	Total time
3:7	No	0.25	159.84	160.09
3:7	Yes	0.39	140.27	140.66
3:10	No	0.25	160.07	160.32
3:10	Yes	0.40	140.49	140.89
4:10	No	0.48	134.33	134.81
4:10	Yes	0.78	120.27	121.05
4:15	No	0.32	151.03	151.35
4:15	Yes	0.51	134.23	134.74
5:10	No	0.49	125.38	125.87
5:10	Yes	0.79	112.45	113.24
5:15	No	0.32	150.02	150.34
5:15	Yes	0.51	132.57	133.08

Figure 8. Timing Data of Model **ball.**

d:h	Bound	Preprocess time	Ray trace time	Total time
5:7	No	3.58	212.80	216.38
5:7	Yes	5.52	187.30	192.82
5:10	No	3.57	203.48	207.05
5:10	Yes	5.50	187.38	192.88
5:13	No	3.55	212.16	215.71
5:13	Yes	5.47	200.94	206.41
6:7	No	6.46	124.05	130.51
6:7	Yes	10.61	113.88	124.49
6:10	No	6.38	129.99	136.37
6:10	Yes	10.49	115.43	125.92
6:13	No	6.26	141.26	147.52
6:13	Yes	10.41	119.64	130.05

Figure 9. Timing Data of Model **ngk.**

Figure 8 and 9 are timing data with different d:h of the model "ball" and "ngk", respectively. Wang and Lai [18] proposed that d (the maximum level of subdivision) and h (the maximum halfspace numbers existed in a cell) can influence the execution time of both creating and ray tracing SDSM. It is of importance to choose an optimal d:h since it can effect ray tracing results. Figure 8 and 9, show some interesting facts. First, we can find out that the ray tracing time using a hybrid-SDSM is faster than using an SDSM only. Second, if we compare the ray tracing time of using a hybrid-SDSM with different d:h, we can find an optimal d:h is 5:10. This shows that to produce the best results we might consider the maximum level of subdivision, the maximum halfspace numbers stored in a cell and using bounding box or not.

5. Conclusions

A hybrid-SDSM structure is proposed in this paper for accelerating ray tracing CSG solids. The fundamental results demonstrate that it accelerates the performance of ray tracing. The first test model performs 25% time reduction, while the second one shows 8% time reduction. It is concluded that the hybrid-SDSM is of success to accelerate ray tracing computing obtaining the benefits of both the spatial subdivision and the bounding volume hierarchy.

Future work is to include model with larger model complexity and to proceed the system analysis to understand the relation of d, h and b.

Reference

[1] A. Appel, "Some techniques for shading machine renderings of solids," In *AFIPS Spring Joint Computer Conference*, pp. 37-45, 1968.

[2] J. Arvo and D. Kirk, "*A survey of ray tracing acceleration techniques*," In A.S. Glassner, *An Introduction to Ray Tracing*, Chapter 6, Academic Press Limited, 1989.

[3] S. Cameron, "Efficient bounds in constructive solid geometry," *IEEE Computer Graphics and Applications*, pp. 68-74, May 1991.

[4] J.H. Chuang and W.J. Hwang, "A new space subdivision for ray tracing CSG solids," *IEEE Computer Graphics and Applications*, pp. 56-62, November 1995.

[5] A.S. Glassner, *An Introduction to Ray Tracing*, Academic Press Limited, 1989.

[6] A.S. Glassner, "Space Subdivision for Fast Ray Tracing," *IEEE Computer Graphics and Applications*, vol. 4, no. 10, pp. 15-22, October 1984.

[7] A.S. Glassner, "Spacetime ray tracing for animation," *IEEE Computer Graphics and Applications*, pp. 60-70, March 1988.

[8] R.A. Goldstein and R. Nagel, "3-D visual simulation," *Simulation*, vol. 16, no. 1, pp.25-31, 1971.

[9] E. Haines, "A proposal for standard graphics environments," *IEEE Computer Graphics and Applications*, vol. 7, no. 11, pp. 3-5, November 1991.

[10] T.L. Kay and J.T. Kajiya, "Ray tracing complex scenes," *ACM SIGGRAPH'86*, pp. 269-278, vol. 20, no. 4, August 1986.

[11] M. Mantyla, *An introduction to solid modeling*, Computer Science Press, 1988.

[12] R.G. Oliver, "*Hierarchical Data Structure for Solid Modeling*," Ph.D. thesis, Department of Mechanical Engineering, University of Leeds, UK, June 1987.

[13] A.A.G. Requicha And J.R. Rossignac, "Solid modeling and beyond," *IEEE Computer Graphics and Applications*, vol. 13, no. 9, pp. 31-44, September 1992.

[14] A.A.G. Requicha, "Representations for rigid solids : Theory, methods and systems," *ACM Computing Surveys*, vol. 12, no. 4, pp. 437-464, December 1980.

[15] J.R. Rossignac, "Beyond solid modeling," *Computer Aided Design*, vol. 23, no. 1, pp. 2-3, January 1991.

[16] S.D. Roth, "Ray casting for modeling solids," *Computer Graphics and Image Processing*, vol. 18, no. 2, pp. 109-144, 1982.

[17] J.N. Spackman, "Scene Decompositions for Accelerated Ray tracing," Ph.D. thesis, University of Bath, UK, February 1990.

[18] C.M. Wang, C.W. Tseng, B.J. Chen and D.H. Lai, "Techniques for fast ray tracing CSG solids," In *Proceedings of the National Computer Symposium Taiwan*, December 1993.

[19] J.T. Whitted, "An improved illumination model for shaded display," *Communications of the ACM*, vol. 23, no. 6, pp. 343-349, June 1980.

[20] G. Wyvill and T.L. Kunii, "Space division for ray tracing in CSG," *IEEE Computer Graphics and Applications*, vol. 9, no. 4, pp. 28-34, April 1986.