INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 9 (5): 956-961, 2010
ISSN 1812-5638
© 2010 Asian Network for Scientific Information

Research on Cube-based Boolean Operation in Cellular Semantic Feature Modeling System

[1]Li-Juan Sun, [1]Ying-Hao Jin and [2]Da-Song Sun
[1]College of Computer Science and Technology, Harbin University of Science and Technology, Harbin, China
[2]Computer Center, Harbin University of Science and Technology, Harbin, China

Abstract: In order to improve the speed and performance of Boolean operation, we propose a Cube-based algorithm for the rapid Boolean operation. And this algorithm creates Cubes for all triangles of the feature entities, takes intersecting detection rapidly by the sufficiency principle of feature interaction, defines the inside and outside of triangles by the values of Cube's vertexes instead of the normal lines used in traditional methods and builds a new entity by classified spaces of Cubes. This algorithm can increase the speed and performance of computing and avoid the errors about holes in the new entity. Through the experiments on computer, it has been validated that this new algorithm is more adaptable and practicable.

Key words: Cube, Boolean operation, cellular model, semantic feature, feature modeling

INTRODUCTION

Boolean operation is one of the basic problems in computer-aided design, computational geometry and computer graphics (Schneider and Eberly, 2004). Although, the concept of Boolean operation was proposed a long time ago and is widely used in a variety of fields as a main way, its algorithms have defects (Rivero and Feito, 2000) and limitations (Kaibo *et al*., 2006).

In modern CAD and CAID systems, some algorithms of Boolean operations request that the objects of Boolean operation are the same type (Liang and Caiming, 2006), some are inaccurate (Peng *et al*., 2005), because they build a new entity by taking an approximation or classification (Mantyla, 1986) and some are complexity and inefficient (Tomas, 1997) especially in complex models (Hongjun *et al*., 2003).

This study mainly discusses the Cube-based Boolean operations in HUST-CAID, a cellular semantic feature modeling system developed by Research Institute of Computer Applied Techniques of Harbin University of Science and Technology. This algorithm can not only increase the speed of Boolean operations, but also overcome defects mentioned above.

CELLULAR SEMANTIC FEATURE MODELING STYTEM

Feature modeling is increasingly being used for modeling products (Bidarra and Bronsvoort, 2000). One of its main advantages over conventional geometric modeling is the ability to associate functional and engineering information to shape information in a product model. Semantic feature modeling is a declarative feature modeling approach. This means that, in contrast to many current approaches, feature specification and model maintenance are clearly separated. All properties of features, including their geometric parameters and validity conditions, are declared by means of constraints. The main advantage of declarative modeling is the freedom in the type of constraints that can be specified and therefore in the way a model can be edited and maintained.

The Cellular model is a non-manifold representation of the feature model geometry, integrating the contributions from all features in the Feature dependency graph. The Cellular model represents a part's geometry as a connected set of volumetric quasi-disjoint cells, in such a way that each one either lies entirely inside a shape extent or entirely outside it. The cells represent the point sets of the shape extents of all features in the model. Each shape extent is, thus, represented in the Cellular model by a connected subset of cells. Furthermore, the cellular decomposition is interaction driven, i.e., for any two overlapping shape extents, some of their cells lie in both shape extents (and are called interaction cells), whereas the remaining cells lie in either of them. As a consequence of this, two cells can never volumetrically overlap. They may, however, be adjacent, in which case there is an interior face of the Cellular model separating them. Such a face can be regarded as having two sides, designated as partner cell faces. A face that lies on the boundary of the Cellular Model has only one cell face (one side), that of

Corresponding Author: Ying-Hao Jin, College of Computer Science and Technology, Harbin University of Science and Technology, Harbin, 150080, China

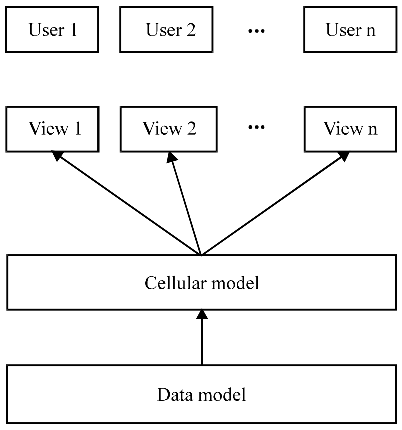

Fig. 1: Structure of HUST-CAID

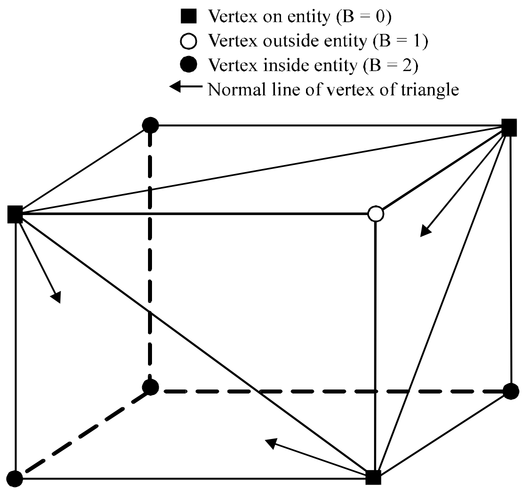
■ Vertex on entity (B = 0)
○ Vertex outside entity (B = 1)
● Vertex inside entity (B = 2)
Normal line of vertex of triangle

Fig. 2: Cube's Vertexes of a triangle

the only cell it bounds. In either case, a cell face always bounds one and only one cell. Each shape face is, thus, represented by a connected set of cell faces.

HUST-CAID is a semantic feature modeling system which bases on the cellular model. It has a structure of three layers as shown in Fig. 1, the bottom layer is a data model which keeps all the information and relations of features designed by users; the middle layer is a cellular model which manages all kinds of elements, shapes and faces of features; and the top layer is a set of views which can provide various information for users in every modeling stage. In the three layers, only the top layer faces users directly and it can get services from the other two layers directly or indirectly. This hierarchical structure can not only keep all the benefits of the semantic feature modeling system, but also make the feature modification not depend on the history trees of modeling by using the Feature Dependent Graph (FDG). Therefore, the CAD/CAID system which uses the cellular model can maintain the semantics of features more effectively and easily during the process of modeling.

MC-BASED BOOLEAN OPERATION

Cubes of entities: Cube is a virtual box surrounding every feature element, which is similar to the Cubes of MC algorithm (Lorensen and Cline, 1987). It can simplify the computing complexity of intersection detection and define the inside and outside face of triangles by its eight vertexes.

To improve the performance of modeling system, Cubes are built automatically for each feature entity. And it is invisible for users. The cube's volume space is determined by the point set of cells which include all the elements of features in the product model and its vertexes can be represented to an ordered pair like (V, B).

If R is a point set of the element of feature entities and Rx, Ry, Rz presents value sets of the projections of R on abscissa, ordinate and vertical axes in the world coordinate system respectively, then $x_1 = \min\{x_i \mid x_i \in Rx\}$, $x_2 = \max\{x_i \mid x_i \in Rx\}$, $y_1 = \min\{y_i \mid y_i \in Ry\}$, $y_2 = \max\{y_i \mid y_i \in Ry\}$, $z_1 = \min\{z_i \mid z_i \in Rz\}$, $z_2 = \max\{z_i \mid z_i \in Rz\}$, then the set of V are as follow: $V_1(x_1,y_1,z_1)$, $V_2(x_2,y_1,z_1)$, $V_3(x_1,y_2,z_1)$, $V_4(x_2,y_2,z_1)$, $V_5(x_1,y_1,z_2)$, $V_6(x_2,y_1,z_2)$, $V_7(x_1,y_2,z_2)$, $V_8(x_2,y_2,z_2)$.

It is very easy to prove that a triangle's vertexes are must on the edges or faces of its cube, whatever its size and direction are.

The parameter B is an integer variables and its value can be 1 when the vertex is outside the surface of the triangle, 2 when the vertex is inside or 0 when the vertex is on the edge or surface of Cube. Thus system will know which side is out and which is in by the values of Cube's vertexes instead of the normal line of the triangle's vertexes (Fig. 2). This method can reduce the computing complexity and computing time greatly.

Detection of intersection: One of the greatest advantages of Cubes is that all their axes are orthogonal or parallel with the world coordinate system's axes, which can make the intersecting detection between any two Cubes of triangles in feature entities become very simple. And the only work for the intersecting detection between two Cubes is calculating whether any vertex of one Cube is inside the other one.

And the Intersection of Cubes is an essential condition of the triangle intersection, i.e., when the triangle intersection happens, the Cube intersection must happen too, otherwise maybe. Thus it can be concluded that if there is no Cube intersection there will be no triangle intersection. So, the new triangle intersection

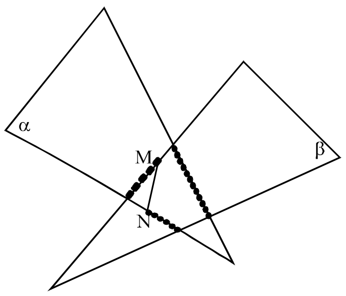

Fig. 3: The intersected triangles

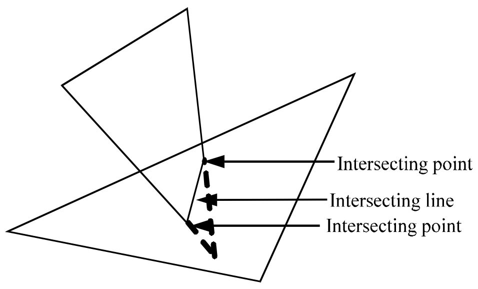

Fig. 4: The divided triangle

consists of two steps with the detection of Cube intersection and the traditional detection of triangle intersection. If there is no Cube intersection, pass triangle detection over directly.

Triangulations of entities: If A and B are two entities relating to the Boolean operation, TA is a set of triangles in A which may intersect with triangles in B and TB is a set of triangles in B which may intersect with triangles in A. These triangles' ID will be stored in a linked list as the same order in entities.

To any triangle ta_i in TA, there is a set P_i in which all the triangles in TB may intersect with it. And P_i must meet the following conditions:

$$P_i \in TB$$

$$P_1 \cup P_2 \cup \cdots \cup P_i \cup \cdots \cup P_n = TB$$

If all the triangles in P_i don't intersect with ta_i, then delete ta_i from TA (because ta_i isn't on the cross section), otherwise record all the intersecting points in ta_i into a linked list L which shared by A and B. When ta_i has been processed, ta_i is deleted from TA and another triangle will be chosen automatically to calculate like ta_i until TA is empty. It is easy to understand that whichever operation (intersection, union or subtraction) it is, if the intersection face of two entities is the same, the linked list L must be the same too (because L is a cyclic list). Since, the intersecting points are on both intersected triangles which belong to A and B respectively, they both belong to A and B too. Thus the last results of L from TA are the same from TB, which makes the procession of Boolean operation become easier.

In Fig. 3, M and N are two intersection points of the triangle α and β and the former is on an edge of β and inside of α, while the latter is on an edge of α and inside of β. The segment MN is the intersection line of α and β, which is part of the whole outline of two entities' intersecting section. Therefore we can get the intersection line by connecting all the segments like MN.

Obviously, in the cellular model, the outline of the intersecting entities' cross section is a closed polygonal line. All of the outline's vertexes are the intersecting points which are inside the triangles of entities and can be calculated easily by the intersection check of TA or TB (In HUST-CAID, the system chooses the smaller one automatically for a higher speed). Therefore, the Boolean operation can be completed easily by simple computation of points in the cellular model instead of the triangle computation which is very complex in old algorithms. And this improvement can enhance the performance of modeling systems based on features largely. Although, the procession referred above is for two triangles, it can be applied directly on the condition of one-to-many or many-to-many triangles intersection too.

The next work after the intersecting detection is to triangulate the intersected triangles, which is crucial in the Boolean operation. In order to avoid errors such as ambiguity surfaces and holes, which may occur in traditional algorithm, modeling systems must triangulate every triangle by using the linked list L which is built in the process of intersecting detection. Here, are the steps:

Step 1: Find all the intersecting points which are inside a triangle and the intersected triangles from L and then compute the intersecting line. If there is not any intersecting point in the triangle, i.e., this triangle is divided entirely by another one, connect the two points directly which are inside the intersected triangles, as shown in Fig 4. Otherwise connect all points according to the order in L

Step 2: Triangulate the triangles. The intersecting lines divide the triangle into two parts which belong to different entities and should be triangulated at the same time. If the two parts are both triangles, there is nothing to do, see a and b in Fig. 5a-f. Otherwise, triangulate the part(s) which is (are) not the triangle, see c, d, e and f in Fig. 5

Step 3: Repeat step 1 and 2, until all the triangles have been processed in TA (or TB)

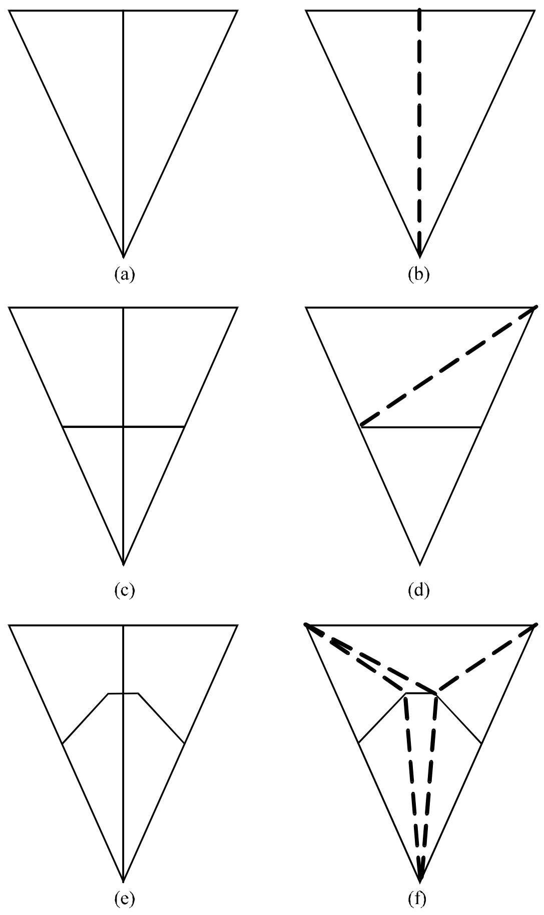

Fig. 5: (a-f) Triangulations of intersected triangles. a, c and e are the original triangles and b, d and f are the triangulated triangles

Classification of spaces and building of entity: The last work of Boolean operation is to build a new entity. For different operation, the new entity may need different triangles, e.g., the intersected part will be remained in the intersection operation or be deleted in the union or subtraction operation. As mentioned earlier, if the triangles intersect, their Cubes must intersect too. And the Cubes divide the volume space of triangles into three parts: SA belonging to entity A, SB belonging to B and SAB shared by both A and B. SA and SB can be determined easily how to process: SA and SB should be remained in union operation and be deleted in intersection operation. And SA should be remained and SB deleted in the operation of A-B; SA should be deleted and SB remained in the operation of B-A.

In SAB, not all parts belong to the new entity. So, we classify the spaces of SAB into four parts: part I is inside both A and B, part II inside A but outside B, part III outside A but inside B and part IV outside both A and B.

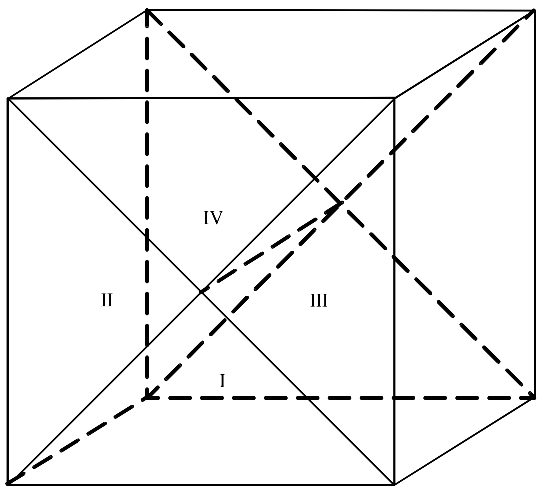

Fig. 6: Parts of SAB. I: Inside A and B, II: Inside A but outside B, III: Outside A but inside B and IV: Outside A and B

It can be proved that the four parts always exist whatever the triangles look like and wherever they are. To understand easily, the Fig. 6 shows a model in which all parts are the same. It is obvious that triangles in part I should be remained and the others should be deleted in intersection operation, triangles in part II and III should be remained and the others should be deleted in union operation and that only triangles in part II should be remained in A-B operation and only triangles in part III should be remained in B-A operation.

In cellular model, all elements including triangles are managed by a data structure-owner list. And both processions-deleting and remaining triangles are achieved by pointers (the handle of triangles in owner list), which can reduce the computing time greatly.

EXPERIMENT RESULTS AND ANALYSIS

Figure 7a-f is experiment results of Boolean operations between two closed entities on HUST-CAID. Figure 8a-e is results between a closed entity and a non-closed entity. Figure 9a-c is between two non-closed entities.

In HUST-CAID, the Boolean operation doesn't be processed as division computation for the consistency of semantics and operations of features when there is(are) the non-closed entity(s). Because the changes are not obvious, there are no results of the subtraction (only A-B) operation between closed entity and non-closed entity in Fig. 8 and the intersection and subtraction operations between two non-closed entities in Fig. 9.

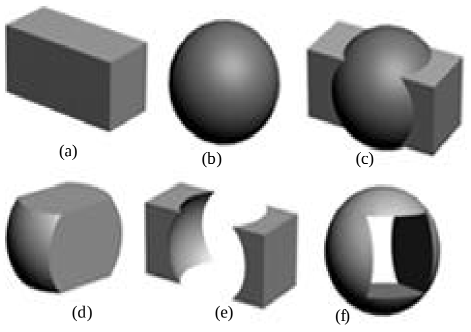

Fig. 7: Boolean operations between closed entities. (a) entity A, (b) entity B, (c) union (A∪B), (d) intersection (A∩B), (e) subtraction (A-B) and (f) substraction (B-A)

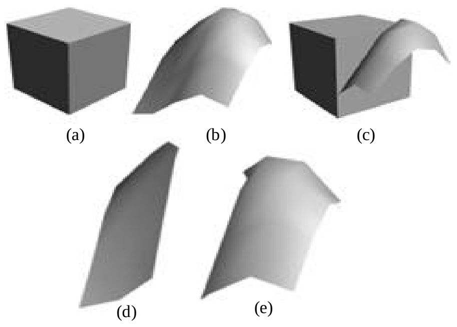

Fig. 8: Boolean operations between closed entity and non-closed entity. (a) entity A, (b) entity B, (c) union (A∪B), (d) intersection (A∩B) and (e) substraction (B-A)

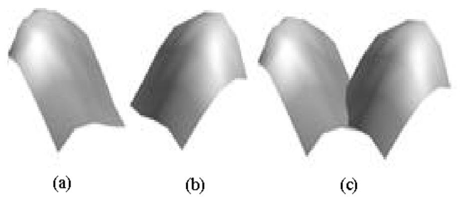

Fig. 9: Boolean operations between non-closed entities. (a) entity A, (b) entity B and (c) union (A∪B)

Table 1: Rates of detecting and triangulating

Triangles of entities	Detected triangles	Triangulated triangles	Hit rate (%)
100	13	12	92.31
200	25	33	88.00
500	67	55	82.90
1000	122	102	83.61

In addition, this algorithm doesn't use any approximation operation and can maintain the accuracy of entities well. Lots of experiments show that the hit rate of selecting triangulated tiangles with this algorithm is very high, even to the complex entities containing 1000 more triangles it can be more than 83% (Table 1).

DISCUSSION

It's easy to know that the efficiency of traditional algorithms of Boolean operation is inversely propotional to the total number of triangles in entities (Jing-yu *et al*., 2006). So, it is inefficient to the traditional algorithms to deal with the complex models (Hong *et al*., 2008). In many CAD/CAID systems, the approximate triangles are used largely instead of quondam triangles for rapid processing (Shiqi and Hongzan, 2005), which may result in unexplained errors (Zhenhua and Yuanjun, 2009), such as holes (Gong *et al*., 2009). However, the new algorithm proposed in this study ignores all triangles which have not any possibility of being triangulated by Cubes. So, there are only a few of triangles remained which need to be checked. It is obvious that the smaller the triangle set which need to be detected is, the faster the speed of intersecting detection is. And whatever entities are, the ratio of detected triangles to all triangles in entities changes a little. So, the Cube-based algorithm of Boolean operation can keep its effecency well.

In addition, the usage of classify of spaces makes the more improvement of efficiency. The new entity created by the new algorithm is exact to the quondam entities and has not any ambiguities or errors.

CONCLUSION

In this study, a Cube-based algorithm of Boolean operation is proposed, which can be used for not only closed entities but also non-closed entities. And this algorithm can improve the speed of Boolean computation greatly without any approximate operation. So, it is more adaptable and practicable. In future studies, the ability of semantic maintaining and the speed of intersection detecting will be improved further.

ACKNOWLEDGMENT

This study is supported by National Nature Science Foundation of China under Grant No. 60173055.

REFERENCES

Bidarra, R. and W.F. Bronsvoort, 2000. Semantic feature modeling. Comput. Aide Des., 32: 201-225.

Gong, Y.X., Y. Liu, L. Wu and Y.B. Xie, 2009. Boolean operations on conic polygons. J. Comput. Sci. Technol., 24: 568-577.

Hong, Y., L. Hao and L. Wenhe, 2008. Basic Boolean operation research in catmull-clark subdivision surface. J. Comput. Res. Dev., 45: 1259-1268.

Hongjun, L., W. Congjun and H. Shuhuai, 2003. Boolean operation for polygon with holes. J. Huazhong Univ. Sci. Technol., 31: 18-20.

Jing-yu, W., Q.I. Yan and W. Yan, 2006. Discrete Boolean operation based on triangular patch. Trans. Shenyang Ligong Univ., 25: 25-28.

Kaibo, G., Z. Lichao, W. Congjun and H. Shuhuai, 2006. Implementation of boolean operations on STL models. J. Huazhong Univ. Sci. Technol. (Nat. Sci. Edn.), 34: 97-99.

Liang, X. and Z. Caiming, 2006. A topology complexity based method to approximate isosurface with trilinear interpolated triangular patch. J. Comput. Res. Dev., 43: 528-535.

Lorensen, W.E. and H.E. Cline, 1987. Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Comput. Graphics, 21: 163-170.

Mantyla, M., 1986. Boolean operations of 2-manifolds through vertex neighborhood classification. ACM Trans. Graphics, 5: 1-29.

Peng, Y., J.H. Yong, W.M. Dong, H. Zhang and J.G. Sun, 2005. A new algorithm for Boolean operations on general polygons. Comput. Graphics, 29: 57-70.

Rivero, M. and F.R. Feito, 2000. Boolean operations on general planar polygons. Comput. Graphics, 24: 881-896.

Schneider, P.J. and D.H. Eberly, 2004. Geometric Tools for Computer Graphics. Publishing House of Electronics Industry, Beijing.

Shiqi, O. and B. Hongzan, 2005. Frame Boolean operation in surface subdivision. J. Huazhong Univ. Sci. Technol. (Nat. Sci. Edn.), 33: 61-63.

Tomas, M., 1997. A fast triangle-triangle intersection test. J. Graphics Tools, 2: 25-30.

Zhenhua, Z. and H. Yuanjun, 2009. Queer problems on 2D Boolean operation. Comput. Appl. Software, 26: 24-33.