Enhancing Sense of Reality by Efficient and Precise Collision Detection in Virtual Environments

Chuan-Jun Su

Department of Industrial Engineering & Management, Yuan Ze University
135, Far-East Rd., Chung-Li, Taiwan, ROC
iecjsu@saturn.yzu.edu.tw

Abstract. As the foundation of user-system interaction in virtual environments, collision detection is a time-consuming process and few real-time interactive algorithms for general objects developed. Most of the existing methods aim for reducing the computation time for some special cases. Collision detection algorithms developed are either not fast enough for practical applications or restricted to a class of specific model. In fact, a general analysis of the performance of collision detection algorithms is extremely difficult because performance is situation specific. The motivation of this work is to satisfy the real-time interaction and high precision requirements of a Virtual Environment (VE) for applications such as virtual design, virtual assembly, virtual training for manufacturing operations and maintenance.

Keywords: Collision Detection, Virtual Environment, virtual reality.

1 Introduction

Collision detection has been extensively used in the fields of robotics operation and process planning for manufacturing. Recently, the research in collision detection has also played an important role in the field of VR. Collision is required specifically to simulate the real-life situations of touching, grasping, moving, and striking. The objective of collision detection in VR is to report accurately all geometric contacts between objects and to respond in real-time. In a VE the user and objects may move with abrupt changes in direction and speed. To provide a sense of reality, a VR system has to efficiently and precisely perform collision checking between virtual objects in a dynamic environment. Due to its widespread importance, there has been an abundance of work on the problem of collision detection. Since Boundary Representation (Brep) and CSG are the most commonly used object representation scheme, the existing collision detection approaches for these two object models are reviewed in this section.

1.1 Collision Detection Approach for Brep-Represented Objects

Spatial decomposition techniques - Octree [1], k-d trees [2], BSP-trees, brep-indices [3], tetrahedral meshes, and regular grids [2] are all examples of spatial decomposition techniques. By dividing the space occupied by the objects, one only

V.G. Duffy (Ed.): Digital Human Modeling, HCII 2007, LNCS 4561, pp. 969–978, 2007.

needs to check for contact between those pairs of objects (or parts of objects) that are in the same or nearby cells of the decomposition. Using such decomposition in a hierarchical manner can further speed up the collision detection process.

Hierarchical BV techniques - One of the fastest general-purpose polygonal model collision detection system is the "RAPID" system, which is based on OBBtrees, implemented by Gottschalk. Lin, and Manocha [4]. The efficiency of this method is due in part to an algorithm for determining whether two oriented bounding boxes overlap. Another of the fastest approaches publicly available for performing collision detection among arbitrary polygonal models is the QuickCD, which is based on BV-trees, developed by [2].

Distance-based techniques - There has been a collection of innovative work which utilizes Voronoi diagrams [5, 6] to keep track of the closest features between pairs of objects and calculate the distance between them. If the distance between a pair of objects goes below the predefined threshold, a collision is then declared [6]. One popular system, I-COLLIDE [5], uses spatial and temporal coherence in addition to a "sweep-and-prune" technique to reduce the pairs of objects that need to be considered for collision. This software works well for many simultaneously moving convex objects.

1.2 Collision Detection for CSG-Represented Objects

There have been very few approaches dealing with collision detection for CSG-represented objects. Zeiller [7] proposed a three-stage method for detecting collisions among CSG objects using S-bounds [8] and space subdivision. The shortcomings of this method are that at each time step in dynamic simulation, the system needs to recompute new S-bounds for the transformed objects and to perform the spatial subdivision for CSG objects to create an Octree-like data structure that is very time-consuming.

In summary, most of the proposed methods aim for reducing the computation time for some special cases. Most proposed collision detection algorithms are either not fast enough for practical applications or restricted to a class of specific model. In fact, a general analysis of the performance of collision detection algorithms is extremely difficult because performance is situation specific. The motivation of this work is to satisfy the real-time interaction and high precision requirements of a VE for applications such as virtual design, virtual assembly, virtual training for manufacturing operations and maintenance.

In this paper, an efficient and precise collision detection algorithm for rigid objects is proposed. The main idea of our approach is that firstly all Brep objects are converted into CSG representation to take full advantages of CSG's tree structure by using BCSG-1.0 [9]. A localization procedure to detect the potential regions of collision is then performed by testing the interference of several 3-D bounding objects with some non-leaf nodes of two objects' CSG trees. Subsequently, a finer search for true collision is performed within that region using simple and convex bounding representation's face information. CSG's "divide-and-conquer" paradigm, decision rules for sub-tree freezing and result evaluating, distance-aided collision detection for BVs using I-COLLIDE package [5] and adaptive BV selection strategy are used in the localization procedure.

2 Pre-processing

2.1 Brep to CSG and CSG to Brep Conversion

To take full advantages of CSG structure in generating efficient algorithm for the collision detection of virtual objects, all Brep objects are converted into CSG representation in the pre-processing stage using BCSG-1.0 [9]. The procedure induces a set of primitives from a given boundary representation that are used to partition the space into Boolean atoms (cells) that classify either in or out with respect to the solid. The union of those atoms that classify in is optimized using geometric and Boolean algebra techniques to obtain efficient (often minimal) CSG representation of the solid. An example of Brep to CSG conversion performed by BCSG-1.0 is shown in Figure 1.

2.2 Hybrid CSG/Brep Object Model Generation

To fully utilize the strength of both CSG and Brep models for the task of collision detection, a hybrid CSG/Brep object model is constructed in the pre-processing stage. In the construction of a hybrid object model, the input CSG represented object is firstly converted into a Brep model and the links between each object face and its corresponding primitive in the CSG-tree are also established during the conversion. Figure 2 illustrates the procedure of the hybrid CSG/Brep object model generation and Figure 3 depicts the structure of the Hybrid CSG/Brep object model.

2.3 BV Candidates Generation

3-D bounding boxes or spheres are used unambiguously for extremely fast but rough detection of object collision because of their simplicity in intersection computation. To achieve not only fast but also accurate localization for the potential collision region, the BV adopted should be as small as possible and as simple as possible. It is desirable to reduce the unnecessary interference checking among CSG tree nodes by using bounding objects.

One effective way is to minimize the parts of overlapping between the BVs of the target solids. A smaller bounding object implies less interference checking while simpler BV simplifies the required computations such as the minimum distance between objects. Therefore, a list of BVs is generated for each branch node in the CSG tree in the preprocessing stage rather than one fixed type of BV. In the detection stage, the smallest BV for a particular detection case is adaptively selected from the list. In the pre-processing stage of the proposed method, the BVs for each non-primitive CSG node object are selected from the following candidates: sphere, cube, wedge, cylinder, cone, and convex hull. These candidates can be generated by using the covariant analysis based principal axis computation [10] and Qhull package [11]. Figure 4(a), (b), (c), and (d) are the examples of the object-aligned bounding box, bounding sphere, bounding wedge and bounding cylinder of the object respectively. Figure 4(e) illustrates the bounding box of one of the intermediate nodes in the CSG tree of the object.

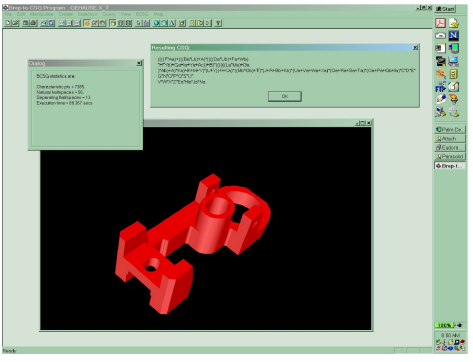

Fig. 1. An example of Brep to CSG conversion by BCSG-1.0

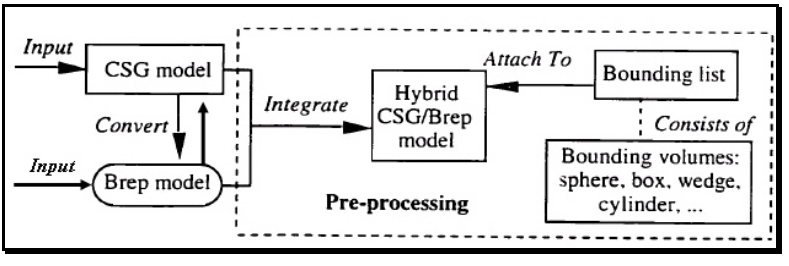

Fig. 2. The procedure of generating the hybrid CSG/Brep object model

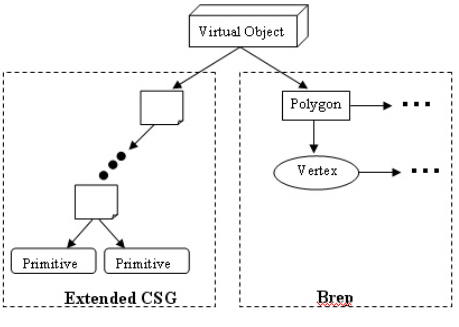

Fig. 3. The Hybrid CSG/Brep object model structure

2.4 BV List Attached to a CSG Tree

Once BV candidates are generated, they are attached to the non-primitive objects in a CSG tree in a list structure called "BV list". The nodes of the BV list are linked in ascending order according to the degree of fitness. Each node in a BV list has four fields:

1. BV type field: ob — object-aligned bounding box: co — bounding cone; cy — bounding cylinder; we — bounding wedge; sp — bounding sphere; ch — convex hull.

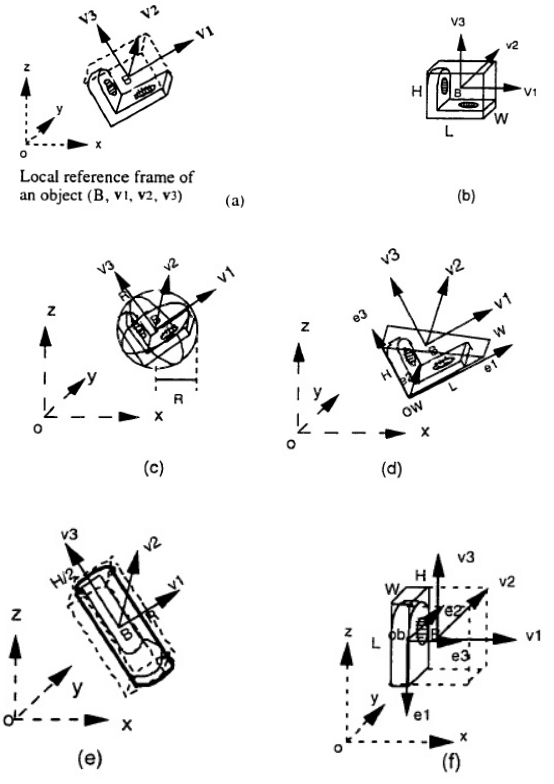

Fig. 4. (a) an example of the basic axis of the object: (B, v_1, v_2, v_3); (b) An example of the object-aligned bounding box of the object A; (c) the bounding sphere of the object A: (d) the bounding wedge of the object A; (e) the bounding cylinder of the object B; (f) the bounding box of one of the intermediate nodes in the CSG tree of the object A.

2. Parameters field: For each type of BV, the corresponding parameters are listed:

(ob, L, W, H; x, y, z; e1, e2, e3): where, L, W, H are the length, width, and height of the bounding box respectively, the point OB(x, y, z) is the central point of the object-aligned bounding box ob, and vectors e1, e2, e3 are the direction vectors of the BV in the local reference system (B, v1, v2, v3);

(co, R, H; x, y, z; e1, e2, e3): where, R, H are the radius, height of the bounding cone: the point OC(x, y, z) is the central point of base circle of the bounding cone co, and vectors e1, e2, e3 are the direction vectors of the BV in (B, v1, v2, v3);

(cy, R, H; x, y, z; e1, e2, e3): where, R, H are the radius, height of the bounding cylinder; the point OY(x, y, z) is the central point of base circle of the bounding cylinder cy, and vectors e1, e2, e3 are the direction vectors of the BV in (B, v1, v2, v3);

(we, L, W, H; x, y, z; e1, e2, e3): where, L, W, H are the length, width, and height of the bounding wedge respectively, the point OW(x, y, z) is the central point of the object-aligned bounding wedge we, and vectors e1, e2, e3, are the direction vectors of the BV in (B, v1, v2, v3);

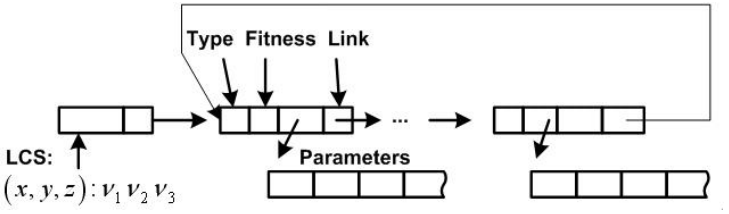

Fig. 5. The data structure of a BV list

(sp, R, x, y, z): where, R is the radius of the bounding sphere sp; the point OS(x, y, z) is the central point of the bounding sphere in (B, v1, v2, v3);

For the convex hull of an object, the parameters are a set of vertices, edges, and facets for the convex hull ch of the object in (B, v1, v2, v3).

3. Fitness degree field: (for simplicity, the volume of BV is stored), and
4. Link field: the pointer linked to next BV node. Figure 5 depicts the data structure of a BV list.

3 Collision Detection Algorithm

After the completion of the pre-processing stage, with the pre-constructed hybrid CSG/Brep object model, a "divide-and-conquer" method can be used to localize the region of interest hierarchically. At each branch node-checking step, it is determined whether there is an intersection between each pair of objects roughly by selecting a most suitable BV from each BV list attached. If no intersection occurs, stop. Otherwise, the CSG operators determine whether further collision checking is needed between children of the current checked node against the current checked node of another object. We devise a set of decision rules for the determination. In this section, adaptive BV selection strategy and decision rules to use the sub-tree freezing approach are presented.

Distance-Based Interference Test for Bounding Objects

Intuitively, if the minimum distance between two solid objects is greater than zero, there is no collision between these two objects. Otherwise, they are touching each other. In this section, we use the distance-based method to detect if two BVs are interfering with each other or not. The suitable BVs of each object are selected from BV lists of each testing object in a CSG tree. In our approach, we have selected sphere, box, cylinder, cone, wedge, and convex hull as BV candidates.

Cohen et al. implemented a package I-COLLIDE for Finding the closest points between two convex polyhedra [5]. It runs almost always in constant time if the previous closest features have been provided and are (on average) linear in the total number of vertices if no special initialization is done. In the membership test process, the running time is proportional to the number of feature pairs traveled. This method was adopted to get the approximate distance between two primitives using the polyhedral representation of each primitive.

If a bounding sphere is selected as a BV from the BV list, a simple and fast method for calculating exactly the distance of a pair of primitives has been derived. A set of

closed form equations for precise distance computation by using their relative positions and geometric features were derived.

Adaptive BV Selection

For all possible pairs of BVs, a term c ($0 < c \leq 1$) is defined to express the relative degree of complexity in computing the distance between two objects in a pair. For example, sphere-sphere pair has the lowest c value because checking the distance between two centers and the sum of two radii can easily test the pair's intersection. The cone-cone intersection test may be a relatively complex one to perform. This pair has the higher c value. The BV selection is based on the criterion of minimizing S which is defined as the product of the degree of complexity c and the degree of fitness of two BVs, f1 and f2, of each non-leaf in each CSG tree. That is,

Min. $S = c \, x \, f_1 \, x \, f_2$

Subject to $fi = 1 - \frac{v_o{}^i}{v_b{}^i}, i = 1, 2.$

Where $v_o{}^i$ and $v_b{}^i$ (i = l, 2) represent the volumes of non-leaf object i and the attached BV i respectively. Before the collision test for two target objects is performed, a bounding volume is adaptively selected for each target object based on this selection criterion. The intersection test between BVs of two branch nodes is then checked.

Decision Rules for Sub-tree Freezing and Detection Result Evaluating

To improve the collision checking for BVs in each CSG tree, sub-tree freezing technique is used to suspend detection of sub-trees (branches). Freezing means that no further actions should be carried out with the "frozen" sub-tree for the object currently being detected at this time step. One branch may be "frozen" or "not frozen" according to the set operation in the node and the detection result of BVs. Detection result of the upper level of node checked can be evaluated. Therefore, decision rules for sub-tree freezing and detection result evaluating are constructed.

Given two objects (or CSG branch nodes) a and b, at the beginning of the procedure of finding collision detection, it can be determined whether there is an intersection between each pair of objects roughly by selecting each BV from each BV list attached to the root of the objects. If no intersection occurs, stop. Otherwise, according to the operator in node a, check first the current level node on the second CSG tree b against the next lower level node(s) on the first CSG tree a. The left child and the right child of node a are denoted as L.a and R.a.

If the operator is UNION, collision between b and L.a is tested. If no collision occurs, checks further the collision between b and R.a. If no collision occurs, stop. If the operator is INTERSECTION, collision between b and L.a is detected first. If no collision occurs, stop; otherwise, check the collision between b and R.a. Table 1 shows the decision rules for union and intersection operation. If the operator is DIFFERENCE, a rough detection between b and R.a is done with BVs. If no collision occurs, check the collision between b and L.a. Otherwise, it is very possible that b is colliding with the complement region of node a. In this time, a polygon intersection test should be executed between the local complement region, b and R.a. If no collision occurs, the result depends on whether there is any collision between b and L.a. Table 2 shows the decision rules for difference operation.

Table 1. Decision rules for union and intersection operators

$UNION(L.a, R.a)$			$INTERSECTION(L.a, R.a)$		
$(L.a, b)$	$(R.a, b)$	(a, b)	$(L.a, b)$	$(R.a, b)$	(a, b)
Yes	Yes	Yes	Yes	Yes	Yes
Yes	No	Yes	Yes	No	No
No	Yes	Yes	No	Yes	No
No	No	No	No	No	No

Table 2. Decision rules for difference operator

$DIFFERENCE(L.a, R.a)$		
$(L.a, b)$	$(R.a, b)$	(a, b)
Yes	Yes	Yes
No	Yes	Yes
Yes	No	Yes
No	No	No

Table 3. Computation time required for each processing cycle of collision detection between two objects with different complexity

Object one (O1)	**Hand**	**Hand**	**Hand**	**Panel**	**Workpiece**	**Probe**	**Probe**
O1 triangles	86	86	86	524	164	1268	3268
Object two (O2)	**Fixture**	**Probe**	**Tool**	**Hand**	**Probe**	**Hand**	**Tool**
O2 triangles	1120	3268	11294	86	1268	186	11294
# of Steps	1000	1000	1000	1000	1000	1000	1000
# of contacts	323	360	320	186	162	356	279
Average time (in msec.)	0.05	0.35	0.21	0.025	0.22	0.25	0.31

Recursive Algorithm for Collision Detection

With the pre-constructed hybrid CSG/Brep object model, a "divide-and-conquer" method, adaptive BV selection strategy and decision rules are used to enable fast and accurate localization of the potential collision regions of each object. We can see that the above procedure is recursive. After several branch node-checking steps, the leaf nodes are checked. For two positive leaf nodes generating union or intersection operations, the distance-aided collision detection approach is used to perform the intersection test. For the complemented part(s) generated by subtraction, all of

corresponding faces in Brep model are found first to quickly rule out unnecessary polygon-based collision checking, and then polygon-based intersection test is used for the remaining faces.

Implementation
The algorithm proposed in this paper was implemented in C on a 200 MHz R4400 CPU SGI Onyx Reality Engine. The Qhull package [11] for computing convex hulls was used to construct convex hulls of the branch nodes and the I-COLLIDE package was incorporated in the scheme. As a general assessment for the algorithm, the average processing time required for detecting collision between two objects in a typical Virtual CNC Training setting is depicted in table 3.

4 Conclusion

We presented an efficient and precise collision detection algorithm for CSG-represented objects in a VE. The objects are assumed to be rigid solids and would not be deformed during and after collisions. In summary, this method has following advantages:

1. Using natural hierarchical CSG tree structure and adaptive selection strategy for bounding volumes, it realizes an effective and efficient intersection or contact region localization. An intersection or contact can be localized into two leaf nodes of each object after limited steps of simple bounding volume intersection test.
2. The hybrid CSG/Brep representation facilitates the combination of the advantages of the faster, less accurate BVs with accurate distance-assisted collision detection and precise polygon-based intersection test.
3. Our methods have been implemented. Effectiveness and correctness have been proven by experiments with various object pairs with different geometrical complexity. For complex object pairs, our method is significantly more effective.

References

1. Moore, M., Wilhelms, J.: Collision detection and response for computer animation. Comput. Graphics. In: SIGGRAPH '88 Proc vol. 22, pp. 289–298 (1998)
2. Held, M., Klosowski, J., Mitchell, J.:Real-time collision detection for motion simulation within complex environments. In: SIGCRAPH'96 Visual Proc. (1996), vol. 151 (1996)
3. Vanecek, G.: A Data Structure for Analyzing Collisions of Moving Objects. In: Proc. of the 24th Annual Hawaii Int. Conf. on Sys. Sci. (1991), pp. 671–680 (1991)
4. Gottschalk, S., Lin, C., Manocha, D.: OBBTree: A hierarchical structure for rapid interference detection. Comput. Graphics 30, 171–180 (1996)
5. Cohen, J., Lin M., Manocha D., Ponamgi M.: I-COLLIDE: An Interactive and Exact Collision Detection System for Large-scale Environments. In: Proc. ACM Interactive 3D Graphics Conf. (1995), pp. 189–196 (1995)

6. Lin, M.C., Manocha, D.: Fast interference detection between geometric models. The Visual Computer 11, 542–561 (1995)
7. Zeiller, M.: Collision detection for objects modeled by CSG. Visualization and Intell. Design in Eng. and Architecture, 165–180 (1993)
8. Cameron, S.: Efficient Bounds in Constructive Solid Geometry, IEEE Computer Graphics & Applications, 68–74 (1991)
9. Hartquist, E.: BCSG-1.0: A Practical Implementation of Boundary to CSG Conversion,Tech. Rep. CPA93-3, Sibley School of Mech. & Aerospace Eng., Cornell Univ. (1994)
10. Wu, X.L.: A linear-time simple bounding volume algorithm. In: Graphics Gems HI, pp. 301–306. Academic Press, Inc, San Diego (1992)
11. Barber, B., Dobkin, D., Huhdanpaa, H.: The Quickhull Algorithm for Convex Hull. ACM Trans, on mathematical Software 22(4), 469–483 (1996)