Using Separating Planes between Objects for Efficient Hidden Surface Removal

Alok Kumar, Vivek Kwatra, Balpreet Singh, Sanjiv Kapoor
Department of Computer Science & Engineering
Indian Institute of Technology
New Delhi

Abstract

This paper describes a new technique for generation of binary space partition trees representing polyhedral objects and outlines an efficient algorithm for operations such as composition, deletion and motion of objects. We assume a modified version of BSP trees with all the polygon fragments lying at the leaf nodes described in [9]. The efficiency of merging algorithm for two BSP trees depends on the separability of the polyhedras represented by them. We aim to achieve efficiency by use of an ellipsoidal cover of polyhedral object to determine the separating plane between two objects. This paper primarily aims at moving from the domain of polygons to that of objects in dynamic environment.

1 Introduction

We consider the hidden surface removal problem for three dimensional scenes composed of polyhedral objects in dynamic environment. Objects are represented as binary space partition trees. An algorithm is presented for rapid updates of the BSP tree used to describe the scene. We allow for composition, deletion and and motion of objects. The algorithm suggested here maintains the elegance of the BSP trees while allowing objects to be defined as arbitrary polyhedras and dealing with them at the same abstraction level. The use of binary space partition trees for representing polygonally described three dimensional scenes finds its root in an observation by Schumaker et. al.[8], in 1969., that separating planes in an environment can be arranged to determine clusters of objects, so that objects in a cluster on the same side of a plane as the viewpoint cannot be obscured by those on the far side.

The BSP tree represents a recursive hierarchical subdivision of n-dimensional space into convex subspaces. This subdivision is achieved by a partitioning hyperplane. Restricting ourselves to a scene described by a set of 3-D polygons, this hyperplane would correspond to a separating plane resulting in *positive* and *negative* half-spaces. The original BSP identified the separating planes with the planes corresponding to polygons in the scene. Such a construction results in a recursive algorithm for determining a back-to-front ordering of polygons by traversing the tree in an in-order manner depending on the viewpoint. Fuchs et. al.[7] suggested a heuristic to reduce the number of fragments resulted during the construction of a BSP tree. Paterson and Yao[6] show that it is possible to construct a tree of size $O(n^2)$, the previous known bound being $O(n^3)$.

Though appropriate for static environments such a structure loses its advantages in a dynamic environment. A new method for dynamically changing BSP trees was investigated by Y. Chrysanthou and M. Slater[1]. They constructed the BSP trees incrementally as done by Thibault and Naylor[2,3]. The restructuring required on motion of an object in Scene tree was accomplished by reinsertion of the modified object into the tree. Since deletion of any polygon would result in two unconnected trees, they were merged incrementally. The complexity of restructuring was claimed to be reduced due to nearness of any moving polygons to the leaf node after the first deletion.

The algorithm described here allows us to move from the domain of polygons to that of objects i.e. polyhedras. Objects are defined and constructed recursively as a polygon or a merge of two objects. Hence, it helps in defining three-dimensional scenes are composed of a set of objects. These objects can be represented, added, deleted and moved independently. An object could possibly be composed of a hierarchy of other objects. The BSP tree identifying an object has all the constituent polygon fragments at the leaf nodes. Binary Space Partition tree representing the complete scene is constructed by successive merging of objects. Merging operation is developed on the work done by Thibault and Naylor[2,4] for representation of arbitrary polyhedral solids and set operations on polyhedra. We give simpler and efficient dynamic updates, especially that of deletion, by using a separating plane between two objects. The separating plane methodology has been described in [9]. This version of the BSP tree was used to delete polygons in a BSP tree in constant time per fragment. It critically used separating planes between pairs of polygons to partition the space. Another separating plane approach has been used by Torres [10]. However, as noted in the subsequent approach in [1], the approach looses simplicity and elegance by using a 6-level tree.

In section *2* we outline the definition and properties of an object. There we describe a new BSP tree construction algorithm, which is also used for merging of two objects. Efficient procedures for deletion and motion of an object are also descibed in the same section. We also outline a methodology, which could be used for determination of separating plane between two objects in section *3*. Conclusions are presented in section *4*.

2 A New Approach to Dynamic Objects

Throughout we deal with an object as the basic unit of operation. It is assumed that objects are represented as collection of polygons forming polyhedra. Our algorithm supports the following aspects of dynamic changes in 3D scenes : -

1. Change of View point
2. Addition of an object into the scene
3. Deletion of object from the scene
4. Composition of objects
5. Motion of an object

We represent objects by a binary space partition tree. Motion of objects is defined as the motion of the corresponding tree. That is, we don't reconstruct the tree for an object on motion, since the structure of the tree is maintained under translations and rotations. All the nodes (planes as well as polygons) are just updated to maintain current orientations. We also allow for a hierarchy of objects, which have a parent-child relationship. Each object is either a polygon or consists of another set of objects. Motion of an object, results in motion of the all its sub-object. When an object moves it is deleted from its parent and then reinserted, propagating the effect to the root object i.e. one which does not have any parent.

2.1 Merging algorithm for objects

Each object has its own BSP tree. When two objects are to be merged, then a new tree representing the combined object needs to be constructed. We devised a strategy in which the efficiency of tree construction is dependent on the ability to determine a separating plane between the two objects. A separating plane **Pl** is defined as a hyperplane, which divides the space into two semi-spaces such that each object lies on the either side of the plane.

Algorithm for merging two BSP trees B_1 and B_2 to generate a new BSP tree **B** is as follows -

Step 0) If the roots of each B_1 and B_2 are plane then Goto 1 else if root of only one of the trees is plane Goto 5 else Goto 6.

Step 1) Determine ellipsoid covers (Cov_1, Cov_2) for each of the objects. Use these covers to determine the separating plane **Pl** between the ellipsoids and hence the objects.(Determination of cover for an object and mid_plane for ellipsoids explained in section 3) If a partition plane could be determined goto Step 2) else Step 3). It should be noticed that it is possible that a partition plane is not determined, even though one existed, since we are approximating the convex hull of an object by an ellipsoid to determine the middle plane. This has been done since if lot of effort in made in establishing the optimal separating plane, then the advantage of mid_plane would be lost.

Step 2) Resultant tree has **Pl** as root and the trees of the original objects as its left and right sub trees. Exit.

Step 3) In case, we are unable to determine separating plane between the two objects, the tree of one of the objects, say B_2, is split using the root plane **Rp** of the other object. The two split trees are such that all polygons in a tree lie on one side of the root plane and the polygons of both trees lie on opposite sides.

Step 4) The splitting plane **Rp** is made the root of the new tree. The merge of left sub tree of B_1 and the left split of B_2 becomes the left child of **Rp**. Similarly, the right child of **Rp** is the merge of right sub tree of B_1 and the right split of B_2 . Exit.

Step 5) In case, the root of one of the trees is not plane then the tree with polygon root is inserted in the other tree. The insertion of polygon into the BSP tree is done using the module described in section 3. Exit.

Step 6) For merging two polygons, we determine separating plane between them using the algorithm described in [9]. This plane is made the root of tree and the polygons are split accordingly.Exit.

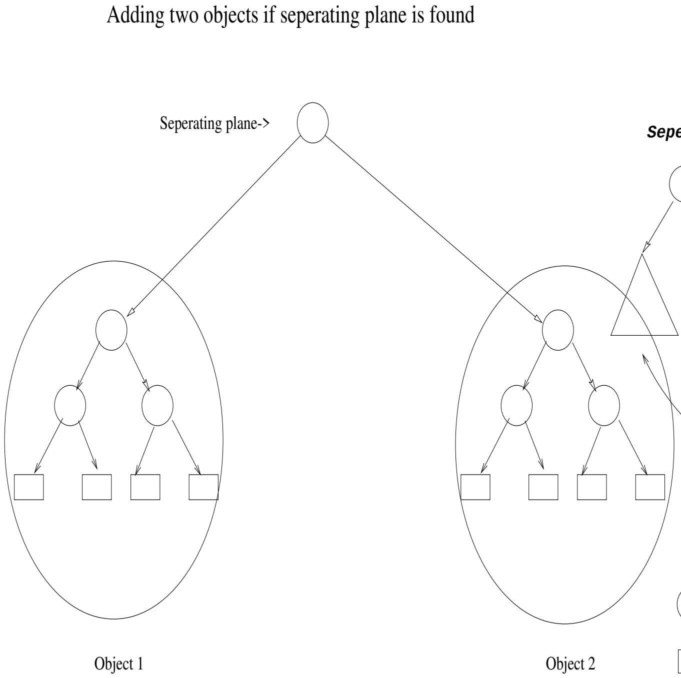

Figure 1: **Addition of object if separating plane is found**

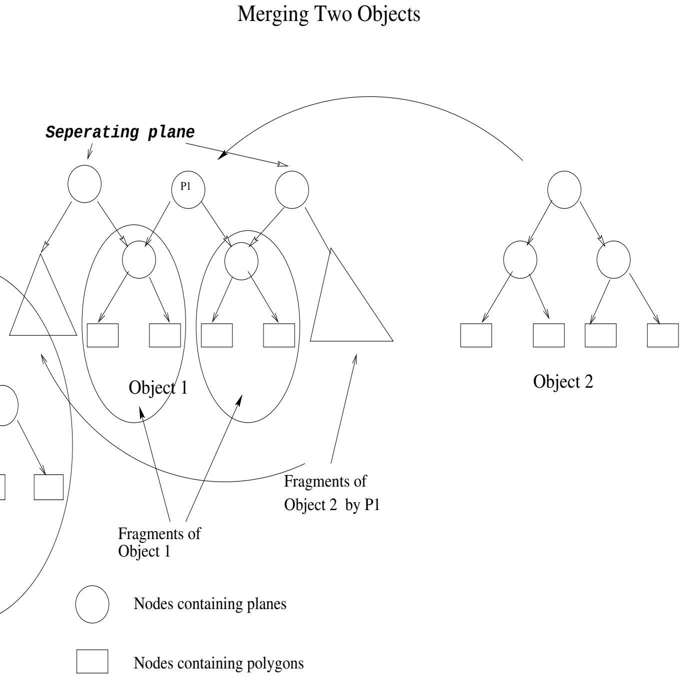

Figure 2: **Inserting one object into another**

Since an object is considered to be made of a polygon or merge of two objects we can construct the BSP tree of an object incrementally by using the above algorithm. That is, by merging a new polygon with already constructed BSP tree.

It can be easily proved that the BSP tree constructed using the above algorithm, on traversal, will result in correct priority determination for a given viewpoint. If we are able to determine a unique separating plane then definitely all polygons of tree B_1 will be on the side opposite to that of B_2, by definition of the separating plane. Otherwise, all polygons in the left/right split of B_2 will lie to left/right of the root of resultant plane. This is because, they are merged with the left and right sub trees of B_1. Since, splitting always reduces the height of B_1 by 1, the base case is insertion of a polygon into a tree hence the BSP tree constructed could be guaranteed to provide correct ordering. Elegance of the algorithm lies in the fact that the tree structure would be preserved for a large number of iterations. Besides, the complexity of the algorithm is related to the complexity of the view. That is. in cases when separating plane can be found the algorithm performs extremely well. Even when, there is a failure, efforts are made to determine the hyperplane at a lower level.

2.2 Deletion of an object from parent object

When an object is deleted, it is also deleted from its parent object, if any and that object is deleted and that added to its parent and it is repeated till we reach an object with no parent. Deletion of an object from its parent is a linear time operation if we maintain with each object a pointer to the first fragment in the tree and all the fragments of the object are linked sequentially within the tree.

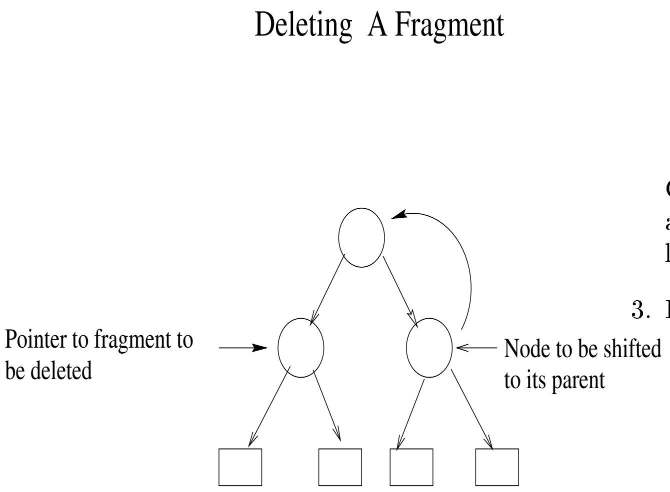

Figure 3: **Deleting a fragment of object**

2.3 Motion of an object

When an object moves the complete BSP tree is updated, though the structure remains same. All polygons along with the partitioning planes are modified (given the same motion as the object). If the object has some child, then that child must also update its tree. Besides, the object is first deleted then added in its immediate parent, propagating the effect to the root object. Though, very much optimised, the motion of an object could result in redundant planes in the parent object, after a number of iterations. This could be checked by periodic reconstruction of the complete structure i.e. after a fixed number of iterations.

3 Determination of separating plane between two objects

We cover an object with an optimal ellipsoid so as to assist determination of separating plane. An ellipsoid consists of three radii R_1, R_2,R_3 in mutually perpendicular directions and a **Center**. Center of the object is found as the center of the minimal cuboidal cover, such that sides of the cuboid are parallel to coordinate axes, of the object. The ellipsoidal cover, then is constructed using the following algorithm -

1. Find the most distant point C_1 of the object from the center of the object. The C_1 - center is chosen as axis and R_1 is assigned its length.

2. For all points, p, apart from C_1

   (a) construct an ellipse with "this" point p and C_1 lying on the ellipse and C_1-center as major axis.

   (b) determine the corresponding length of minor axis.

   Choose the point C_2 with maximum minor axis length and fix corresponding minor axis length as R_2.

3. For all points, p, apart from C_1 and C_2

   (a) construct an ellipsoid with "this" point p, C_1 and C_2 lying on the ellipsoid and **Center** as its center.

   (b) determine the corresponding length of the third axis. Choose the point C_3 with maximum length of third axis and fix corresponding third axis length as R_3.

Hence, an ellipsoid covering an object is determined. In case, one of the object is a polygon only , we do not find its ellipsoidal cover and we find separating plane between the ellipsoidal cover of the other object and the polygon, which is done by transforming the ellipsoidal cover to sphere and then to find separating plane between a sphere and a polygon , we can find plane perpendicular to the line, perpendicular to the plane of the polygon extended and passing through the center of the sphere. The separating plane is found, such that it lies midway between the sphere and the polygon.

Otherwise, we need to determine separating plane between two ellipsoids (E_1,E_2) in arbitrary orientation. We transform one of the ellipsoids, say E_1, into a sphere by compression of the axes. The other ellipsoid E_2 is altered using the same transformation matrix. Now, we bring E_2 to the origin with its axes aligned with the coordinate axes. So as to find the separating plane, a point P_2 is selected on E_2, such that the line **L** joining P_2 to the center of transformed E_1 is normal to E_2. The line **L** is also normal to E_1(transformed) and intersects E_1 at P_1. Mid_plane is selected as perpendicular to **L** at the mid_point of segment($P_1 - P_2$).

Let coordinates of P_2, which is unknown, be (x,y,z) and center of E_1(known) be (u,v,w) and length of axes of E_2(known) be (a,b,c). The equations derived from conditions under which a line is simultaneously normal to E_1 and E_2 are :-

1. $(a-b)xy - avx + buy = 0$
2. $(b-c)yz - bwy + cvz = 0$
3. $(a-c)xz - cuz + awx = 0$
4. $ax^2 + by^2 + cz^2 = 1$

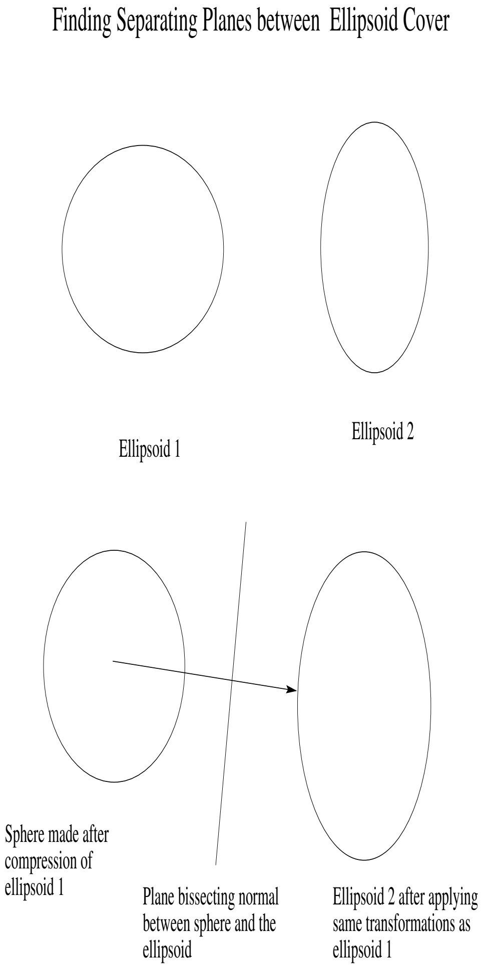

Figure 4: **Finding Separating plane using ellipsoid**

One of the equations of the first three has to be eliminated as they are not indepedent , then we get three equations and three variables. As there are three polynomial equations and three variables, a solution exists, hence we will always be able to find a solution of above equations.

References

[1] Chrysanthou, Y., and Slater, M., Computing dynamic changes to BSP trees, Computer Graphics Forum (EUROGRAPHICS '92 Proceedings), 11(3), 321–332, sep 1992.

[2] Naylor, B., Amanatides, J., and Thibault, W., Merging BSP Trees Yields Polyhedral Set Operations, Computer Graphics (SIGGRAPH '90 Proceedings), 24(4), 115–124, aug 1990.

[3] Naylor, B., Interactive solid geometry via partitioning trees, Proceedings of Graphics Interface '92, 11–18, may 1992.

[4] Naylor, B., Partitioning tree image representation and generation from 3D geometric models, Proceedings of Graphics Interface '92, 201–212, may 1992.

[5] Naylor, B., SCULPT An Interactive Solid Modeling Tool, Proceedings of Graphics Interface '90, 138–148, may 1990.

[6] Paterson, M., and Yao, F., Efficient Binary Space Partitions for Hidden-Surface Removal and Solid Modeling, Discrete and Computational Geometry, 5(5), 485–503, 1990.

[7] H.Fuchs, Z.M. Kadeem and B.F. Naylor. On visible surface generation by a priori tree structures. ACM Computer Graphics, 1980.

[8] R. Schamacker, B.Brand, M.Gilliland and W.Sharp. Study for applying computer generated images to visual simulations. US Air force Human Resources Lab, Air Force Systems Command, Brooks AFB, TX, September 1969.

[9] A. Kumar, V. Kwatra, B. Singh, S. Kapoor, Dynamic Binary Space Partitioning for Hidden Surface Removal, ICVGIP'98

[10] Torres, E., Optimization of the Binary Space Partition Algorithm (BSP) for Visualization of Dynamic scenes. Computer Graphics Forum(EUROGRAPHICS '90 proceedings), 9(3):507-518, 1990.