Volumetric Particle Separating Planes for Collision Detection

by

Brent M. Dingle

Fall 2004
Texas A&M University

Abstract

In this paper we describe a method of determining the separation plane of two objects during collision detection. This method is specifically designed for volumetric particle representations of objects, volpars. This particular technique offers an advantage over previous methods because it only looks for a separating plane for the areas local to the point of contact. This increases the robustness of the technique as it will require no special considerations for concave versus convex cases. This should be similar in behavior to those methods based on nearest features within Voronoi regions.

1 Introduction

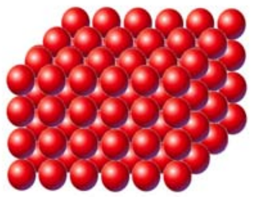
In this paper a method for determining the separating plane between two objects in collision will be presented . This method is specifically designed for volumetric particle representations. The advantage this method offers is its robustness. The fundamental concept of the method is in finding a separating plane between two colliding objects that is based on the local geometry near the point of collision. Thus the separating plane might not entirely separate the objects, but separates the surfaces (faces) of the objects that are in collision. This allows the method to function regardless of the convexity (or concavity) of the colliding objects. It is believed that this algorithm when used in conjunction with "large" bounding box techniques will offer a time performance that is competitive with any other collision detection method currently used. It is also believed that further time reductions may be possible if the techniques are adapted to take advantage of the GPU. This is conceivably possible due to the simple structure of particles and the nature of collisions between them. The necessity of demonstrating this equivalent performance is driven by the desire to encourage the usage of volumetric particle representations in physically based modeling.

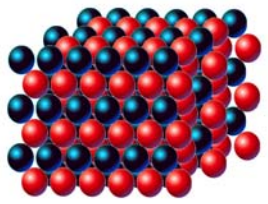
This paper will begin with a description of what a volumetric particle representation, volpar, of an object actually is. It will then give a brief description of the "general" collision detection process. From there it will present the new method and a simple example. The paper will conclude with a brief summary and possible future directions and usage of the method.

2 Volumetric Particle Representations

Volumetric particle representation of objects is best described as a representation of an object created by filling the object with particles and joining them with some form of connector. This particular representation is a modification of that used in fluid and cloth simulation. In those cases the former has no connectors between the particles and in the latter the connectors are usually springs. For the sake of this paper we will assume the connectors are rigid rods that allow no motion between the particles. However the method to be described shortly should function on any object representation that is similar to a volumetric particle representation, volpar.

Two examples of volpar representations of boxes are illustrated below:





It should be noted that while volpars are similar to voxels, they are not the same thing. Voxels are (almost) always cubes and have no explicit connections between them. Whereas volpars usually have an explicit connection defined and are most often spheres. Another major difference is that voxels were designed for display purposes, whereas volpars are designed for simulation purposes. However, in a sense, voxels may be considered a subset of volpars.

3 Collision Detection in General

For this paper collision detection will be thought of as a task necessary to perform collision response. More specifically, we will not just seek to determine *if* objects are in collision, but *how* and *where* they are in collision. Thus we effectively have two processes: collision detection and collision response. While we will only give details concerning the former, we must do so in a way that is useful to the completion of the latter.

Almost all collision detection techniques can be divided into two phases: the quick, broad phase and the more accurate, and time consuming narrow phase. The broad phase usually only determines which objects *might* be in collision. The narrow phase will determine if two objects are indeed in collision and the details of that collision. These details usually include which two objects are involved, the point of contact, the normal of the separating plane (or equivalent) and the type of contact [Bar1999].

In the broad phase the simplest routines partition space and report objects that are in the same space as potentially colliding [Ove1992]. Other routines give every object in the world some form of bounding box or sphere. In these techniques you have Axis Aligned Bounding Boxes (AABB) and Object Oriented Bounding Boxes (OOBB) [Got1996]. With these boxes, various hierarchal tree structures are created which allow for quickly determining if two objects might be in a collision state. In other collision detection methods nearest features of any two objects are maintained throughout the simulation and are used to detect when objects might collide [Lin1991, Mir1998]. Most of these routines are efficient and effective in implementation. However when the actual collision is narrowed down and the collision point is determined they still must determine a separating plane (or equivalent) to perform the collision response.

This paper is not suggesting anything better for the broad testing phase of collision detection. The methods currently used can easily be used with volpars. However there is an advantage in using the volpar representation in that it can calculate the separating plane of two objects as fast, if not faster than any method designed for any other representation. It also allows the automatic processing of features local to the area of collision and in this sense is similar to the Voronoi, nearest features techniques.

4 Method of Determining the Separating Plane

For this method we will assume we have used a good routine to determine that two objects are in collision and we know the approximate point of initial contact. The task we are left to determine is to find a plane that sufficiently separates the two objects so that a collision response may be performed. We make no assumptions about the objects being in the same form or shape as they were at the beginning of the simulation. We also make no assumption that the number of faces of an object remains constant. Thus this method should work for deformable or non-rigid objects as well as rigid bodies.

In a very real sense this method is based on that used in determining the separating plane of rigid convex objects. In that scenario an exhaustive search and comparison of all faces of both objects will succeed in finding a separating plane. We will take a similar approach, but must determine a way to define the faces we wish to explore.

All collisions between two volpar objects will be a collision between multiple particles of the two objects. Effectively this gives three cases: there is one, two, or more than two contact points between the two objects. In each case we will first derive a set of possible normals for the separating plane. We will then test each possibility.

In the case where there are three or more contact points, we know there are three or more unique particles in at least one of the two objects. Assuming those three particles are not collinear then the potential separating plane is the cross product generated by the two vectors formed from those three particles. This amounts to a face-face or vertex-face collision. If the three particles are collinear then the other object must have at least two unique particles involved in the contact points. This is likely an edge-edge collision and the cross product of the vectors formed by taking two unique particles from either object will give a possible separating plane. As an aside, it is also possible to find potential normals of separating planes by using the contact points themselves.

In the case of two contact points, at least one of the objects must have two unique particles involved in the contact. If the other object also has two unique particles involved in the contact then a possible separating plane is generated by creating two vectors, **v1** and **v2**. This is done by setting **v1** to be the vector created from the difference of the centers of the particles in the first object and **v2** is likewise created from the centers of the particles in the second object. If one of the objects, say the second, only has one unique particle involved then **v2** is formed by taking the difference between the actual contact points. Once both **v1** and **v2** have been determined then a possible separating plane is generated from their cross product.

In the case of one contact point the two objects must temporarily be advanced a small time step forward until more than one contact point occurs. Thus allowing the application of the above described methods. This may experience the same type of difficulties that arise in vertex-vertex collisions, however that will not always be the case as this may also describe a vertex-face or vertex-edge collision.

In all cases each possible separating plane must be tested to see that it actually separates all the particles of each object involved in the collision. This test is a simple derivation of the plane equation defined by the normals calculated above and assuming the plane must go through a corresponding contact point.

This may also include a small predetermined number of neighbors of each particle involved in the collision. This allows for the locality of the separating plane to be validated. For convex objects it should be noted that the derived separating plane will separate all the particles of one object from all the particles of the other object.

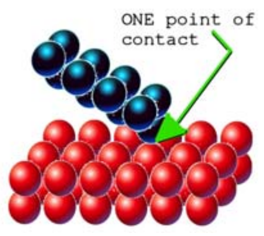
It should be immediately recognized that the repeated calculation of vector products is a time consuming process. In actual implementation of this method it would be advantageous to maintain a list of surfaces for each particle in each object. This list would remain constant until the object breaks apart or deforms. So in reality most of the calculations would be done before the simulation begins. The list of surfaces would likely be a global list of normals with each particle holding an index or pointer to the correct normal for the surface of which it considers itself as being a part. The significant point of the above is that it allows everything to be dynamically determined. So even if the list of normals is precalculated there is no assumption that it cannot change and using the above methods it can by dynamically updated.

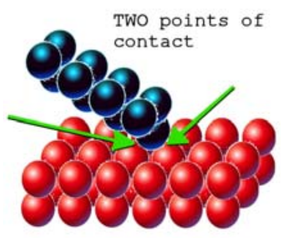
5 Examples

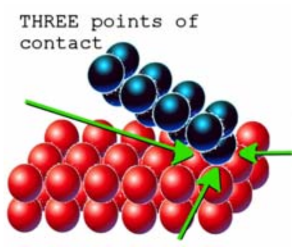
In this section we will present some examples of collision detection using the standard rigid-body terminology and relate them to how a system using volpar representations of bodies would identify them. These examples will include vertex-face, edge-edge and edge-face. From these three examples of collision detection it can be deduced how the other various types of collisions can be detected and resolved.

5.1 Vertex-Face Example

A vertex-face collision in a volpar representation has three unique detection scenarios in the number of particles involved: one to one, one to two, or one to three. Any scenario involving one to more than three can be reduced to the case of one to three.







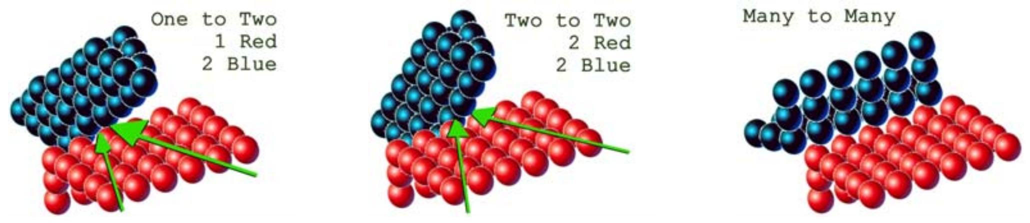
In the event of only one contact point between the two objects is found then the easiest solution is to temporarily move both objects slightly forward in time without making any adjustments. This will likely cause the scenario to change to two or more contact points.

For two contact points we will have one particle in object A and two particles in object B involved in the contacts. To determine a possible separating plane we will create two vectors. The first vector is formed from the two contact points (both are on or near the surface of all three particles). The second vector is created from the center of the two particles of object B. Taking the cross product of these two vectors will give us a possible normal for a separating plane. We will assume the plane must go through at least one of the contact points. With the calculated normal and the stated assumption we can easily form the equation of the possible separating plane. We then test if this plane separates the particles involved in the contact points. We might also test a small number of the neighboring particles of those involved in the contact points.

In the case of three, or more, contact points we will have one particle in object A and at least three particles in object B. We will again form two vectors. To do this we select one of the particles in object B, the center of this particle will be the tail of both our vectors. We then select two other particles in object B. The center of these two particles will be the heads of our two vectors. The possible separating plane will have a normal equal to the cross product of these two vectors. We would then derive the equation for the possible separating plane as described in the case of two contact points and perform a similar test to determine if indeed it is a separating plane.

5.2 Edge-Edge Example

For an edge-edge collision to happen there are also three unique detection scenarios in the number of particles involved: one to two, two to two, or two to three. In the last case the three particles will be roughly collinear. Any case involving more than three particles in either object will reduce to this last case. The co-linearity of the points will distinguish this case from a surface to whatever collision.



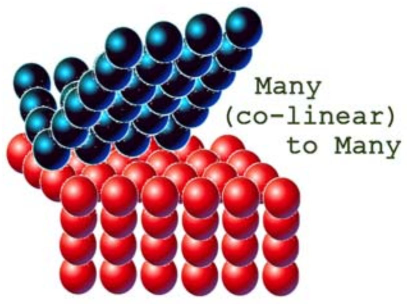
For the one to two particle case we will have two contact points. We will assume the singular particle is in object A and the other two particles are in object B. We will create a vector from the contact points and we will create another vector from the center of the two particles in object B. We then take the cross product of these vectors and use the result as a possible normal for a separating plane. Then as in the vertex-face scenarios we will verify that the plane with that normal through at least one of the contact points is indeed a separating plane for the particles of either object local to the area of contact.

For the two to two particle scenario, we have two particles from either object from which we can form two vectors. The first vector is formed from the center of two particles in object A and the second vector from the center of two particles in object B. Taking the cross product of these vectors gives us a normal for a separating plane that would be tested as described above. It should be noted that we might also use the contact points themselves for one of the possible vectors.

In the two to three, or many collinear to many collinear, case it is easiest to temporarily advance the objects a small time step. This will yield a scenario allowing for a better estimate of the normal of the separating plane. Effectively we will treat the particles of object A as an edge. We then see what particles of B are involved in the contact after the time step forward, excluding those defining the edge of object B. This should amount to an edge-face collision.

5.3 Edge-Face Example

For an edge-face type collision to occur there are two unique detection scenarios: two to three or three collinear to three. Any case involving more than three particles from either object will reduce to the latter of the two scenarios.



In the two to three scenario we have two particles from object A and three non-collinear points from B. Effectively the three center points from the particles of B will define a face. To determine this face we will create two vectors from the particles of B. This will be done by selecting a particle in B to be the tail of both vectors. The other two particles in B will then be the heads of the two vectors. We then take the cross product of these two vectors to establish the normal of the possible separating plane (effectively a face of B). Then, assuming the plane must go through at least one of the points of contact between the two objects we test to see that the plane does indeed separate the particles of A from the particles of B.

In the three collinear to three scenario we do exactly the same procedure as was done in the two to three scenario. We use the non-collinear particles in B to determine a possible separating plane and then we test to make sure the plane separates the particles involved in the contact points.

6 Conclusion

So from the above it should be apparent that the methods of collision detection based on the volpar representation of objects are equivalent to any already existing technique. It should also be apparent that the techniques offer an increase in robustness in collision detection as they are strictly based on the locality of the contact points between objects.

While the calculation of cross products may be a time consuming operation, many of the scenarios could offer pre-computed normal information for each particle. This could be done in a fashion that dynamically responds to the interaction between objects as objects break and deform due to collision. This would be done as a limited update and would not likely consume much time as it would only be necessary in cases where the objects colliding actually alter each other’s shape.

In sum the use of volpar objects would not cause any significant loss in performance due to collision detection. In fact such use may offer more robust collision detection in a simple and easy to implement fashion.

Bibliography

[Bar1999]	Baraff, David and Witkin, A. and Course Notes “An introduction to Physically Based Modelling.” SIGGRAPH 1999 Course Notes.
[Got1996]	S. Gottschalk, M. Lin, and D. Manocha, “OBB-Tree: A hierarchical structure for rapid interference detection,” Proc. ACM Conference SIGGRAPH 96, pp 171-180, 1996.
[Lin1991]	M.C. Lin and J.F. Canny, “Efficient algorithms for incremental distance computation,” IEEE Conf. Robot. Autom., pp. 1008-1014, 1991.
[Mir1998]	B. Mirtich, “V-Clip: Fast and robust polyhedral collision detection,” ACM Trans. Graph., 17:177-208, 1998.
[Ove1992]	M.H. Overmars, “Point location in fat subdivisions,” Inform. Proc. Lett., 44:261-265, 1992.