3rd Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2006)
C. Mendoza, I. Navazo (Editors)

The use of Tetrahedra to Detect Collisions

[1]F. A. Madera, [2]A. M. Day and [3]S. D. Laycock

University of East Anglia, School of Computing Sciences
Norwich NR4 7TJ, UK

[1]f.madera@uea.ac.uk, [2]amd@cmp.uea.ac.uk, [3]sdl@cmp.uea.ac.uk

Abstract

We present an algorithm for collision detection by filling the free space between the objects involved. We compute the closest regions of a pair of objects, and update them during the running simulation. The algorithm is fast because the process depends on the number of neighbouring features of a vertex, the region, and not on all the features of the object. Furthermore, our algorithm is accurate because it deals directly with the features of the objects. This work deals with convex objects, and forms the first part of the general approach for deformable objects.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling

1. Introduction

The algorithm takes into account the mesh connectivity of the objects to walk along the neighbouring faces of the mesh. Once we get a pair of objects, we compute their closest features; then we use these features to determine the occurrence of a collision. The main approach is to explore the Free Space (FS) of the scene using tetrahedra to construct an efficient collision detection algorithm, such as shown in Figure 1.

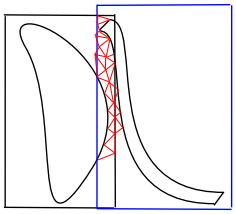

Figure 1: *Tighter triangulation is required in the common region of the bounding volumes.*

2. The Algorithm

Inspired by Agarwal et. al [ABHZ02] who constructs a planar subdivision to detect collisions, we propose to extend the approach to three dimensions. Let $\{O_1, O_2, ...\}$ be a set of objects, where object i is represented as a triangular mesh with $F^i = \{f_1^i, f_2^i, ...\}$ as the set of faces and $V^i = \{v_1^i, v_2^i, ...\}$ as the set of vertices. The collision between objects i and j happens if and only if $O_i \cap O_j$ exists. We construct tetrahedra by using some features of the two objects to fill the free space and compute the volume of them in order to detect the intersection.

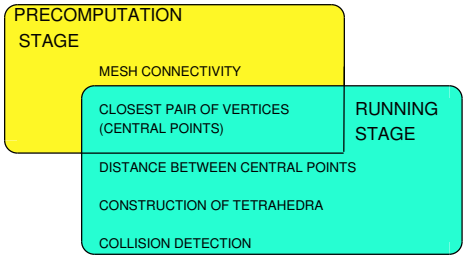

Figure 2: *The routines of the two stages of the algorithm.*

A face f_k of object i is defined as $f_k^i = (v_a^i, v_b^i, v_c^i)$, then we say that this face is a neighbour of its vertices. We use the concept of region to have a control of the closest features.

Definition 1 A **region** R is defined as the neighbouring faces

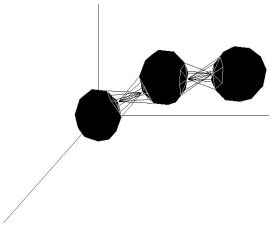

Figure 3: *Tetrahedra formed from the closest features of three objects.*

of a vertex, and is represented as $R(v_c^i, F_{(c)}^i)$, where $F_{(c)}^i$ is the set of neighbouring faces of vertex v_c^i.

In Figure 2, we see the processes of the algorithm, divided in two stages. The precomputation stage works before the animation starts. The mesh connectivity computes the regions of the objects with a $O(|F^i||V^i|)$ time. Among these regions we need the central region, to define the closest features between a pair of objects.

Definition 2 A **central region** of object i with respect to object j, denoted $CR_j^i(v_c^i, F_{(c)}^i)$, is the closest region to object j. (Figure 3).

When the animation starts, central regions can vary as objects move, taking into account the euclidean distance between the vertices of the central regions. This process is cheap because it depends on the number of vertices in the region, and not in the number of features of the object. When the distance between central points is a very small number almost zero, the construction of tetrahedra starts in order to make the contact determination between the features of the object.

3. Experiments

The methods and routines considered to test the algorithm are shown in Table 1.

Stage	Bounding Vol. Method	Free Space Method	Hybrid Method
Precomp.	BVH	Neighbours	Neighbours
Broad	Octree 54		Octree 59
Narrow	BVH 32	Tetrahedra 118	Tetrahedra 43

Table 1: *Number of Collisions of the three methods used.*

We handled 20 polyhedral objects in the scene with vertices ranging from 42 to 170 and faces that go from 80 to 320. The distance step used was 0.2 for motion and the number of time steps was 10,000. The bounding volumes considered were spheres and AABBs. We observe that the FS method detects more collisions than the Bounding Volume approach used in [MDL06]. We addressed more experiments using more steps and increasing the number of objects. Also we changed the distance step from 0.2 to 0.1 and used more complex objects.

The FS method requires more time in the first stage as expected, and the hybrid method takes a long time when the narrow phase starts. This means that the broad phase used in the hybrid method should be improved by using another technique.

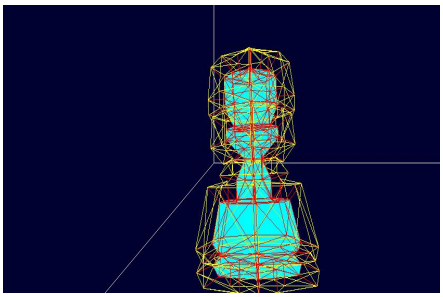

Figure 4: *Extruding the object's mesh outwards.*

4. Conclusions

Our algorithm answers several queries, such as how far apart the objects are, which pairwise are considered, which features are in the close proximity, what the contact determination was, and when the collision occurs.

This method has been proved to work with convex objects. To extend it we are working with concave objects filling the concave regions, converting the object into a convex one, and then using the algorithm for convex objects. Figure 4 shows an object in the first phase of this conversion by extruding the mesh outwards.

References

[ABHZ02] AGARWAL P., BASCH J., HERSHBERGER J., ZHANG L.: Deformable free-space tilings for kinetic collision detection. *International Journal of Robotics Research 21*, 3 (2002), 179–198. 121

[MDL06] MADERA F., DAY A., LAYCOCK S.: Collision detection for deformable objects using octrees. In *Theory and Practice of Computer Graphics* (Middlesbrough, UK, 2006), EG. 122