Particle-based Collision Detection

M. Senin [1], N. Kojekine [2], V. Savchenko [3] and I. Hagiwara [2]

[1] Moscow Institute of Physics and Technology,Kerchenskaya str., house 1"A", building 1, Moscow 113-303, Russia
[2] Faculty of Engineering, Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku, Tokyo 152-8552, Japan
[3] Faculty of Computer and Information Sciences, Hosei University, 3-7-2 Kajino-cho Koganei-shi, Tokyo 184-8584, Japan

Abstract

In the paper we present a novel algorithm for collision detection between complex geometric objects represented by polygonal models and undergoing rigid motions and deformations. Most algorithms described in the literature deal with rigid bodies and are based on some kind of hierarchical representations. We present the alternative approach. The algorithm relies on the idea of "sensor particles": interacting particles distributed on a surface. Two types of particles that interact in a special way are used for determining the minimum distance between two models. The algorithm has been implemented and used in real–time simulation of dynamic interaction between geometric objects. A detailed description of the algorithm, animation examples, and benchmarks are included in the paper. A potential application of this software algorithm is collision detection for animation of bodies with deformable surfaces.

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

1. Introduction

Collision detection has been a fundamental problem in many areas such as physics–based modeling, computer simulated environments, computer animation and robotics. In general the goal of collision detection is to report when a geometric contact occurs. The problem has been well studied in literature. A good survey of collision detection problem can be found in [1].

There are different types of surface representation and collision queries, depending on the application. Models can be represented as constructive solid geometry (CSG), implicit or parametric surface and so on. The surface of a body can be static or can undergo deformations. In this paper, we consider polygonally represented bodies with deformable surfaces. Bodies of this kind often appear in computer animation. These include articulated clothing, biological structures, and other soft or elastic objects.

In the simplest case, it is required to know whether two models are touching. For example, this is useful when we want to arrange bodies in a virtual environment. In computer animation and physics–based simulation it is necessary to know the separation (minimum Euclidean distance) between bodies. Our algorithm can be used to solve all the problems mentioned above.

The collision detection of polygonal objects has been widely studied. Fast and robust methods have been proposed (see for instance [2, 3, 4] and [5]).

Most of the algorithms described in the literature deal with rigid bodies and are based on some kind of hierarchical representation that should be pre–computed before the simulation starts. These algorithms become inefficient in the case of deformable bodies, because an hierarchical representation should be updated after each simulation step. New algorithms have been proposed recently to deal with deformable bodies (see for instance [6], [7]). All such algorithms present new types of hierarchical representation which can be efficiently updated during body deformation.

In this paper we present an alternative approach. Instead of creating and updating a hierarchical representation, we use a set of interacting particles located on a surface. Two types of particles moved under the action of attractive and repulsive forces are used. We can exploit the distribution of these particles to perform collision queries and describe one of the possible implementations of the algorithm.

We clearly understand the limitations of the proposed technique. We should admit that the results of collision detection cannot be sufficiently precise to be used for CAD applications. Nevertheless, high accuracy of collision detection is not required in many other cases. For example, it is not a significant issue for such applications as computer games. As our experiments show, the algorithm provides reasonable accuracy of collision detection by using only a few particles. For sufficiently smooth concave surfaces with over 10000 polygons, 3–5 particles constantly evolving in the collision detection process prevent an object from moving through an obstacle.

The rest of the paper is organized as follows. The next section gives an overview of the collision techniques. In Section 3 we present a detailed description of a particle–based collision detection algorithm. Section 4 contains notes about software implementation of the proposed algorithm. In Section 5 we show examples of animation and speed benchmarks. Section 6 contains conclusions and overview of the future work.

2. Previous Work

The collision detection problem has been widely studied in the literature. Various techniques have been proposed in order to speed up the intersection tests between body pairs.

Bounding volume hierarchies seem to be a very efficient data structure for rapid collision detection of rigid bodies. These include spheres[2, 8, 9], Axis Aligned Bounding Boxes (AABBs)[10, 11], Oriented Bounding boxes (OBBs)[12], k-DOPs[5, 13], Quantized Orientation Slabs with Primary Orientations (QuOSPOs)[14], and spherical shells[15]. The mentioned methods deal with rigid bodies. In the case of deforming bodies, they are not so useful, since these methods rely on pre–computed data which cannot be efficiently updated in real time. There are many papers on collision detection, but less work has been done on deformable bodies. Optimized hierarchical method[16] also exists, and it is more effective both for rigid and deforming objects.

Fast continuous collision detection method, that works well for rigid bodies, was also introduced recently in the paper by Redon et al.[17].

Several other approaches are based on the idea of bounding representation, that approximately models an object with simple primitives such as rectangles. Efficient algorithms (see, for example, [18]), determine if collision has occurred between the bounding representation and only then components of the original model are examined.

A good overview of the animation and collision detection problems can be found in book [19]. A survey [20] presents the state of the art in animation of non–rigid bodies. In particular, models based on the physical theory of elasticity in continuous media and discrete mechanical models that integrate discrete mechanical components are observed. This survey demonstrates some optimization techniques for specifying constraints on the behavior of objects and collision detection and response algorithms.

A general collision detection method for deformable objects has been proposed in [21]. This method deals with triangle soups freely moving in space. However, the high performance of this method breaks down when the overlap region is large and includes many geometric primitives. Another approach has been suggested in [7]. AABB trees are built for each model. The affected nodes in the trees should be updated when a model is deformed. Improvement of this algorithm can be found in [6].

Some new bounding volume hierarchies have been proposed recently. For example, Restricted BoxTrees[22] demonstrate very good construction time and also good time in overlap test.

Some other algorithms are also available for flexible objects[23, 24]. Some methods are designed for bodies undergoing polynomial deformations[25].

Let us notice that, in spite of their effectiveness, standard techniques are expensive in the sense of required memory to store the hierarchical structure; such game consoles as SEGA Dreamcast, Sony PlayStation–2, and others are very powerful, but they are typically limited in memory. The objects that are traditionally used in animation examples have time–dependent but plain surfaces. Detailed surfaces can contain a lot of polygons and have memory limitations for using the hierarchical structures.

Our algorithm exploits the idea of tracking the closest features of bodies. There are various methods[26, 27] based on this approach. These methods use pre–computed Voronoi regions or treat a body as a convex hull of a point set and operate on simplices defined by subsets of these points. These methods are inefficient in the case of deforming bodies.

The 1990s witnessed a growing interest in the development of algorithms based on so-called particle systems, in particular for nonphotorealistic rendering[28, 30] and for resampling of polygonal surfaces[29].

We study the opportunities to use particle systems by looking at simulation of dynamic interaction between rigid bodies with time–dependent polygonally defined surfaces in 3D space. The desired number of particles are spread on the surfaces, and their positions are equilibrated by using mutual repulsive and attractive forces. Then the distance between particles is used to identify the collision area.

3. Algorithm

3.1. Assumptions

At this stage of our project, we are applying our approach to models that satisfy the following geometric and deformation conditions:

- Each model has a surface that is for the most part plain.
- Face's adjacency information is available, i.e., there is a list of neighbor faces for each model point. If not, it can easily be created.
- Adjacency information does not change during deformation. This means that bodies do not change their topology.
- The model remains plain for the most part and the number of convex parts does not increase greatly during deformation.

All these assumptions are valid for various types of models used in animation and computer games.

3.2. Particles

We use two types of particles, by analogy with electric charges. Particles of the first type are placed on the first model; particles of the second type are placed on the second model.

3.3. Interaction

Interaction pursues two different purposes. First, particles of different types should attract each other in order to move to the closest features. This can be done by creating attractive forces between particles of different types.

Second, there should be some interaction, to prevent particles of the same type from gathering in one region of a model. We add a repulsive force between particles of the same type.

Radial attractive and repulsive forces act on particles. The total force $\vec{F_i}$ which acts on a i–th particle is the vector sum of the forces emanating from all other particles:

$$\vec{F_i} = \sum_{i=0}^{n} f(i,j,r_{ij})\vec{r}_{ij}.$$

Unlike the law for electric charges, the laws for repulsive and attractive forces can be different. Repulsive and attractive laws can also be different for different types of particles. We use two functions, f_r and f_a, for repulsion and attraction accordingly:

$$f_r(r) = \frac{1}{r^2},$$

$$f_a(r) = \begin{cases} 0, & r \geqslant R_{eff} \\ \frac{1}{r^2}, & r < R_{eff} \end{cases},$$

where R_{eff} is the effective radius of attraction. R_{eff} should be approximately equal to the linear dimensions of the model.

In general, the initial positions of particles only affect the time of the first collision query. Therefore, it is not very important where particles are located initially. In practice, random distribution works well enough.

3.4. Movement of particles

In the simplest case, particles can be located only at vertices. In this case, we can easily calculate the potential of forces in every neighboring vertex and move a particle to the vertex where the potential has the minimum value.

This technique can be generalized for cases when the particle can be located on edges and faces. We need coordinates of surface points in the neighborhood of a particle. These coordinates can be obtained from adjacency information. Also accuracy of collision detection can be improved by performing additional exact check for faces in particles neighborhood, i.e. in the area where collision is very probable. The size of this neighborhood should depend on sizes of faces which contain vertex particle located at.

3.5. Collision query

Tracking separation between models can be done efficiently, by tracking the distance between particles, as long as we assume that the number of particles is at least equal to the number of convex parts of the model. In practice the number of particles should be twice as much as number of convex parts of the model. This guarantees that separation between models will be tracked efficiently. So the number of particles needed can be determined automatically by analyzing model geometry[27].

At every collision query we turn on an interaction between particles and wait for the particle system to relax. In practice particles occupy some places and movements stop rather soon (this situation corresponds to some local minimum of the system energy). Two numbers, N_{max} and N_{min}, are used to control a particle system to reach almost relaxed state. N_{max} is the maximum total number of interaction steps and N_{min} is the minimum number of moving particles. Therefore, we wait until either the number of particles moved in interaction step is less then N_{min} or the number of interaction steps becomes equal to N_{max}.

Then we can measure distances and find the closest particles of different types (this pair can also be efficiently tracked during interaction). If the distance between these closest particles is less then some tolerance distance, the exact test should be performed in order to find the intersection of particles' neighbor faces. However, many practical tasks do not require such an exact test to be performed.

Collision query of every next simulation step can be performed very fast, if the positional relationship changes slowly. The particle–based approach allows us to capture the temporal and spatial coherence between successive checks easily, even for deformed models.

The values of N_{min} and N_{max} depend on the model's geometry and number of particles n placed on the model. N_{max} should be approximately equal to the number of vertices between particles placed on the model, i.e. the maximum

length of the possible shortest routes between any two particles on the model. Our experiments show that about 10% of particles, in average, are placed near equilibrium position. By this means N_{min} should be about 10 times smaller then number of particles n.

The complexity of performing collision query is $O(n^2)$, where n is the number of particles. Since under above–mentioned assumptions the number of particles is essentially smaller than the number of polygons, this complexity allows us to perform fast collision queries even for models with a large number of polygons.

However, the performance can be noticeably improved, by using caching on the initial steps. If positional relationship changes slowly, the influence of distant particles remains almost constant. Thus we can efficiently cache this influence, and we do not need to perform time–consuming calculations on each iteration.

4. Software implementation and collision response

The proposed algorithm has been implemented in C on top of the OpenRM[31] scene graph.

The particles interactions and bodies' movements can be implemented either in a single thread or in two separate threads. Additional performance improvement can be achieved on multiprocessor systems in the second case.

For models' deformations we use compactly supported radial basis functions (CSRBF) as proposed in[32]. This allows us to define local smooth deformations simply by using a moderate number of control points.

A general solution for the collision response of two arbitrary rigid bodies involves solving a set of 15 linear equations in 15 unknowns, as described in [23, 33]. In our software implementation we use another approach, which is simple and effective in terms of the calculation time. For the sake of simplicity, we do not take into account the velocities of bodies' surface points concerned with deformation. We use a notion of effective mass and pulse to calculate the collision response. Let m be mass of the body; I_{cx}, I_{cy}, I_{cz} — the moments of inertia in the central main axis coordinate system ($oxyz$); $\mathbf{V}$ — the velocity of the center of mass in the world coordinate system ($OXYZ$); $\mathbf{w}_c$ — the angular velocity in $oxyz$; $\mathbf{n}$ — the normal in $OXYZ$; $\mathbf{n}_c$ — the normal in $oxyz$; and $\mathbf{r}_c$ — the radius–vector to the collision point in $oxyz$.

Denote $[\mathbf{r}_c \times \mathbf{n}_c]$ as $\mathbf{l}_c = (l_{cx}, l_{cy}, l_{cz})$. Then the effective mass can be found as

$$m_e = \frac{m}{1 + \frac{ml_{cx}^2}{I_{cx}} + \frac{ml_{cy}^2}{I_{cy}} + \frac{ml_{cz}^2}{I_{cz}}}.$$

The moment velocity along $\mathbf{n}$ is

$$\mathbf{u} = -((\mathbf{V}, \mathbf{n}) + ([\mathbf{w}_c \times \mathbf{r}_c], \mathbf{n}_c))\mathbf{n}.$$

Thus, the effective pulse is

$$\mathbf{p} = -\mathbf{n}\frac{m((\mathbf{V}, \mathbf{n}) + ([\mathbf{w}_c \times \mathbf{r}_c], \mathbf{n}_c))}{1 + \frac{ml_{cx}^2}{I_{cx}} + \frac{ml_{cy}^2}{I_{cy}} + \frac{ml_{cz}^2}{I_{cz}}}.$$

The pulses after collision can be found as

$$\mathbf{p'}_1 = \frac{(m_{e1} - m_{e2})\mathbf{p}_1 + 2m_{e1}\mathbf{p}_2}{m_{e1} + m_{e2}},$$

$$\mathbf{p'}_2 = \frac{(m_{e2} - m_{e1})\mathbf{p}_2 + 2m_{e2}\mathbf{p}_1}{m_{e1} + m_{e2}}.$$

Finally, we can change the velocity and angular velocity as

$$\mathbf{V}' = \mathbf{V} + \frac{\Delta\mathbf{p}}{m}.$$

$$w'_x = w_x + \frac{L_{cx}}{I_{cx}},\ w'_y = w_y + \frac{L_{cy}}{I_{cy}},\ w'_z = w_z + \frac{L_{cz}}{I_{cz}},$$

where $\Delta\mathbf{p}_c$ is the applied pulse in $oxyz$, and $\mathbf{L}_c = [\mathbf{r}_c \times \Delta\mathbf{p}_c]$.

These results were deduced from laws of conservation of energy, momentum and angular momentum for rigid bodies. The complete proof of the above equations is omitted.

5. Results

We tested our software implementation on different polygonal models (obtained for simplicity from implicit models of the alphabet letters).

Figure 1 shows two polygonal models of the letter "T", defined by 1380 polygons and the distribution of 3 particles (shown red and blue) on each model. We found that even a very small number of particles (2–3) is enough to define a sufficiently correct collision point.

We tested the performance of our systems by producing various CSRBF deformations. Four frames from the our test animation using two polygonal models of the letter "M" (model defined by 2968 polygons, CSRBF deformation defined by 14 control vectors) and letter "V" (model defined by 1764 polygons, CSRBF deformation defined by 12 control vectors) are shown in Figure 2.

The most important benchmark for our system is the time needed to perform collision query. This time depends on several parameters, including the size of the model, the number of particles, the speed of models' movement, and the distance between models. For example, for sufficiently small models (two polygonal models of letters "T", 1380 polygons, 3 particles on each model), the maximum time needed to perform collision detection query was 11 ms., and the average time on active interval was less than 1 ms. For larger models (two polygonal models of letters "W", defined by 11920 polygons, 5 particles on each model), results are shown in Figure 3. These tests were performed on a PC with an Intel Pentium 4 2.53 Ghz processor.

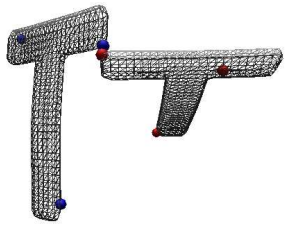

Figure 1: *Illustration of particle distribution.*

Figure 2: *Illustration of collision detection between two models undergoing CSRBF deformations.*

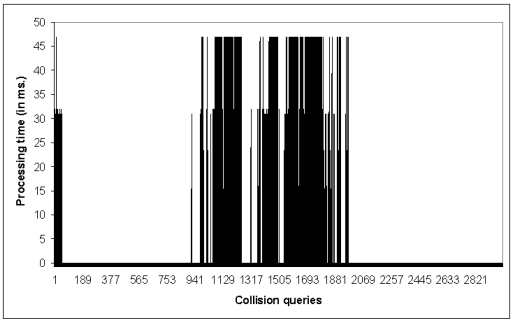

Figure 3: *Benchmarks. Collision detection between two polygonal models of letters "W", defined by 11920 polygons, with 5 particles on each model. The average time needed to perform a collision detection query on active interval (queries nos. 950–1984) was 12.67 ms., and the maximum time was 47 ms.; average time during the "warming" steps (queries nos. 1–49) was 12.06 ms., maximum time was 47 ms. also.*

6. Conclusions

A new method based on using particle systems for collision detection has been presented. The method is used for simulation of dynamic interaction between rigid bodies with time–dependent polygonally defined surfaces in 3D space. The experimental results presented in the paper show that the proposed algorithm can handle sufficiently complicated animation situations in real time.

At the current stage of the project, we have studied only the simple case of collision between two bodies; our future plan includes generalization of this technique for several bodies and other improvements. We have described one of the possible implementations of the algorithm; nevertheless, there are no limitations for combining the particle–based approach with existing collision detection techniques for reliable and precise searching of collision points for face intersections.

The algorithm was implemented in C and tested on a PC under Microsoft Windows, Linux, and FreeBSD operating systems. Implementation is simple and can be easily adapted for special environments (game consoles, etc.); performance can be improved by using special hardware instructions.

The proposed technique is entirely novel, some points are still unclear and require further research; however, we believe that the proposed technique for collision detection can draw further attention from the CG community.

References

1. M. C. Lin and S. Gottschalk. Collision detection between geometric models: a survey, *Proceedings of IMA Conference, Mathematics of Surfaces VIII*, 1998. 1

2. P. Hubbard. Approximating polyhedra with spheres for time-critical collision detection, *ACM Transactions on Graphics (TOG)*, **15**(3):179–210, 1996 1, 2

3. S. A. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition, *In Computer Graphics Forum*, **20**:500–510, 2001. 1

4. M. C. Lin and D. Manocha. Efficient contact determination in dynamic environments, *International J. Comput. Geom. Appl.*, **7**:123–151, 1997. 1

5. G. Zachmann. Rapid collision detection by dynamically aligned DOP-trees, *Proceedings of IEEE Virtual Reality Annual International Symposium*, 90–97, 1998 1, 2

6. T. Larsson and T. Akinine-Möller. Collision detection for continuously deforming bodies, *Proceedings of Eurographics, short presentations*, 325–333, 2001 1, 2

7. G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees, *Journal of Graphics Tools*, **2**(4):1–14, 1997. 1, 2

8. I. Palmer and R. Grimsdale. Collision detection for animation using sphere-trees, *Computer Graphics Forum*, **14**(2):105–116, 1995. 2

9. S. Quinlan. Efficient distance computation between non–convex objects, *Proceedings of IEEE International Conference on Robotics and Automation '94)*, 3324–3329, 1994. 2

10. J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi. I-COLLIDE: an interactive and exact collision detection system for large-scale environments, *Proceedings of the Symposium on Interactive 3D Graphics*, 189–196, 1995. 2

11. M. Held, J. T. Klosowski, and J. S. B. Mitchell. Evaluation of collision detection methods for virtual reality fly-troughs, *Proceedings Seventh Canadian Conference on Computational Geometry*, 205–210, 1995. 2

12. S. Gottschalk, M. C. Lin, and D. Manocha. OOBTree: A hierarchical structure for rapid interference detection, *ACM Computer Graphics (Proc. of SIGGRAPH'96)*, 171–180, 1996. 2

13. J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowrizal, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs, *IEEE Transactions on Visualization and Computer Graphics*, **4**(1):21–36, 1998. 2

14. Taosong He. Fast collision detection using QuOSPO trees, *Proceedings of the Symposium on Interactive 3D Graphics*, 55–62, 1999. 2

15. S. Krishnan, A. Pattekar, M. Lin, and D. Manocha. Spherical shell: A higher order bounding volume for fast proximity queries, *In Proceedings of WAFR'98*, 287–296, 1998. 2

16. S. Bandi and D. Thalmann. An adaptive spatial subdivision of the object space for fast collision of animated rigid bodies, *Proceedings of Eurographics '95*, 259–270, 1995. 2

17. S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies, *Computer Graphics Forum (Proc. of Eurographics '02)*, **21**(3): , 2002. 2

18. D. Baraff. Rigid body simulation, *SIGRPAPH '92 Course Notes)*, **19**, 1992. 2

19. D. H. Eberly. 3D game engine design, *Morgan Kaufmann Publisher*, 2001. 2

20. M.–P. Gascuel and C. Puech. Dynamic animation of deformable bodies, *From object modelling to advanced visual communication*, pp. 118–139, 1994. 2

21. A. Smith, Y. Kitamura, H. Takemura, and F. Kishino. A simple and efficient method for accurate collision detection among deformable polyhedral objects in arbitrary motion, *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 136–145, 1995. 2

22. G. Zachmann. Minimal hierarchical collision detection, *Proceedings of the ACM Smposium on Virtual Reality Software and Technology*, 121–128, 2002 2

23. M. Moore and J. Wilhelms. Collision detection and response for computer animation, *ACM Computer Graphics (Proc. of SIGGRAPH '88)*, **22**(4):289-298, 1988. 2, 4

24. A. Joukhadar, A. Scheuer, and Ch. Laugier. Fast contact detection between moving deformable polyhedra, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 1810–1815, 1999. 2

25. M. Hughes, M. Lin, D. Manocha, and C. Dimattia. Efficient and accurate interference detection for polynomial deformation, *Proceedings of Computer Animation*, 155-166, 1996. 2

26. B. Mirtich. V-Clip: Fast and robust polyhedral collision detection, *ACM Transaction on Graphics*, **17**(3):177–208, 1998. 2

27. S. Cameron. Enhancing GJK: Computing minimum penetration distances between convex polyhedra, *Proceedings of International Conference on Robotics and Automation*, 3112-3117, 1997. 2, 3

28. E. Akleman. Implicit painting of CSG solids, *Proceedings of CSG '98 Set-Theoretic Solid Modeling: Techniques and Applications*, 99–113, 1998. 2

29. G. Turk. Re–tiling polygonal surfaces, *Proceedings of SIGGRAPH '92*, 1992. 2

30. A. P. Witkin and P. S Heckbert. Using particles to sample and control implicit surfaces, *Computer Graphics*, **27**(3):269–277, 1994. 2

31. OpenRM scene graph library. http://openrm.sourceforge.net 4

32. N. Kojekine, V. Savchenko, M. Senin, and I. Hagiwara. Real-time 3D deformations by means of compactly supported radial basis functions, *Proceedings of Eurographics EG2002, short papers*, 35–43, 2002. 4

33. J. K. Hahn. Realistic animation of rigid bodies, *ACM Computer Graphics (Proc. of SIGGRAPH '88)*, **22**(4):299–308, 1988. 4