Comput. & Graphics Vol. 17, No. 4, pp. 379–385, 1993
Printed in Great Britain.

0097-8493/93 $6.00 + .00

A SURVEY OF TECHNIQUES FOR SIMULATION OF DYNAMIC COLLISION DETECTION AND RESPONSE

V. V. KAMAT
Department of Computer Science and Technology, Goa University, Goa-403 202, India

Abstract—When several objects are moved about using computer simulation, there is a chance that these objects will collide with each other, which in turn may result in effecting further change in the movement. The two main issues involved are those of detecting that a collision has occurred and then subsequently depicting a response as an effect of collision. Typically the collision detection is a geometric interference problem that depends on the spatial relationship of objects while collision response is a dynamics problem which involves predicting behavior according to physical laws. In this paper we discuss collision detection and response in general and survey several of the more important solutions proposed and currently in use.

1. INTRODUCTION

In the recent past, some of the most natural and graceful motion in computer animation has been achieved by simulating the behaviour of objects using the laws of Newtonian dynamics. When several objects are moved under the influence of forces, they are likely to interact with other objects. If no special attention is paid to object interaction, these objects will move through each other producing unrealistic and disconcerting visual effects. Whenever two objects attempt to inter-penetrate each other, it is called as a collision. One of the primary requirements of physically based animation systems is to automatically detect the collision. The other related issue is to automatically suggest the motion of objects immediately following collision, which is known as the collision response.

At the very core, collision detection is a spatial interference problem that has been extensively studied in the fields of computational geometry and robotics[1]. In computational geometry the problem is solved for a static environment, *i.e.*, given two objects one has to determine whether the objects intersect. The emphasis is on complex object shapes and exact intersection computation. In robotics, on the other hand, even though the problem is posed in terms of a dynamic environment, *i.e.*, given two objects and their paths determine whether the objects will come into collision during the motion along their paths, the problem can be transformed into static interference detection by using the swept volume technique. In physically based dynamic simulation, however, the problem is of a somewhat different nature. Here the paths of the objects are not known in advance. Rather these paths depend very much on the interaction of these moving objects with other objects in the environment. This interaction can be primarily abstracted in the form of time varying forces acting on each object, which depends on the geometric configuration of the objects in the environment. A straightforward method of collision detection involves the solution of a sequence of static problems one per time step. Although in principle each problem of the sequence can be solved separately, using the algorithms from computational geometry or robotics, these algorithms are not very efficient for use in dynamic simulation. Some of these algorithms solve the problem in more generality than necessary for computer animation. Others do not easily produce the collision points and normal directions that are necessary if collision response is to be modelled[2]. Hence, algorithms have been specially designed for collision detection and response generation taking into account the following facts:

1. In physically based motion simulation, the objects do not penetrate each other but they do touch, hence, it is the contact point that is important and not the exact intersection.
2. Collision response is by and large perceived through the knowledge of the real world experience and can be approximated to the extent that it looks real.

A typical simulator control involving collision detection and response loop does a *step and check* (see Fig. 1). If two bodies are found to interpenetrate, the simulator backtracks to the point in time immediately before the occurrence of the inter-penetration. The time t_0 at which the two objects first come into contact is found using regular root finding method like *regula falsa* or *bisection* method[2]. Once a colliding configuration without inter-penetration is achieved, the contact and normal at contact point are determined between all the contacting bodies. Finally using the penalty method or an analytical method, correct contact forces and impulses are determined. These forces and impulses are then applied and a new time step is begun.

2. COLLISION DETECTION

Here we shall consider the collision detection problem with different types of objects. We shall start with the simplest object, *i.e.*, a particle and then discuss techniques for more complex objects in increasing order of shape complexity.

2.1. *Collision detection for particles*

This technique is the simplest of its kind, where a particle with a point mass is checked for penetration against a plane or a surface. To detect a collision with a plane one simply needs to check the sign of the expression $(x - p) \cdot \hat{n}$ where $\hat{n}$ is the normal to the plane, p is any point on the plane and x is the position of the particle (see Fig. 2).

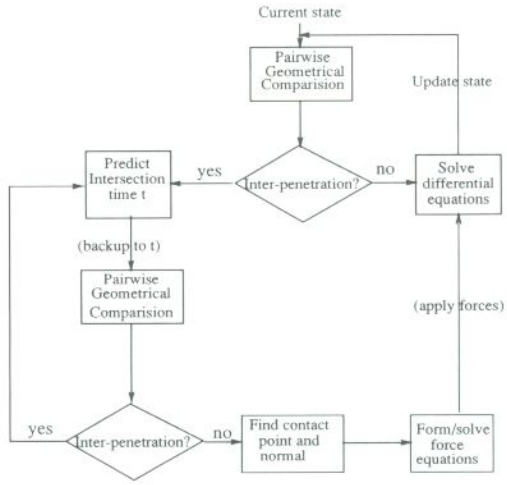

Fig. 1. Simulator control flow.

The particle-particle collision detection is done in two steps. First the distance between the two particles is checked. If the distance is less than the tolerance, then it is checked whether two particles are approaching each other along the line joining the two. If both these conditions hold true, then the particles are declared to be colliding.

2.2. *Collision detection for polygons*

Polygon-polygon collision detection is performed by testing for the penetration of each vertex point of one polygon through the plane of the other and by checking that no two edges intersect. However, in many cases merely testing points versus polygons produce acceptable results. Moore *et al.*[2] and Lafleur[7] use this technique to detect collision between flexible surfaces. Two cases are considered depending on whether the surface polygon is fixed or moving.

When the surface polygon is fixed [in this case a triangle with vertices (P_0, P_1, P_2)], the parametric vector equation is given by

$$P + (P' - P)t = P_0 + (P_1 - P_0)u + (P_2 - P_0)v$$

where P and P' are the beginning and ending position of the point, u and v are the parametric variables for the plane defined by the triangle and t is a time variable for the simulation step. By solving this system for t, u, v one determines whether the point has intersected the triangle during the time step by checking the conditions $0 \le t \le 1$, $u \ge 0$, $v \ge 0$ and $u + v \le 1$ (see Fig. 3).

When the surface polygon is moving, the following parametric vector equation is setup:

$$P + Vt = P_0 + V_0 t + ((P_1 - P_0) + (V_1 - V_0)t)u + ((P_2 - P_0) + (V_2 - V_0)t)v$$

Where P is the point with velocity V per time step, V_0, V_1, V_2 are the respective velocities of the points P_0, P_1, P_2. If we eliminate u and v from the three component equations, we get a fifth order polynomial in t, which can be solved numerically. Each value of t thus arrived at is used to get values for u and v by back substitution, and then the standard $0 \le t \le 1$ and $u \ge 0$ and $v \ge 0$ and $u + v \le 1$ tests are used to determine whether a collision has occurred.

To minimize the computational costs, a hierarchical method involving octree and bounding box tests have been suggested[2]. These tests ensure that a more geometrically precise collision detection test is invoked only when the objects are close to each other and there is a likelihood that they may collide. These enhancements only test the necessary condition for interference and are not sufficient.

2.3. *Collision detection for convex polyhedra*

Although we shall only discuss algorithms to detect collision among convex polyhedra, it is presumed that with some preprocessing the concave polyhedra can be decomposed into a collection of convex ones before applying any of these algorithm. The most naive algorithm checks the face of each polyhedron against the faces of other polyhedra and vice versa, which is computationally $O(N^2)$.

If two polyhedra inter-penetrate, it is almost always the case that either a vertex of one polyhedron is inside the other or an edge of one polyhedron has intersected a face of the other.

A single inter-penetration test between two objects A and B is done in two phases. In the first phase, a test is performed to see whether object B is inter-penetrating object A. This in turn is done in three steps.

1. Test the presence of vertices of B inside of A. Each vertex of B is compared to every face of A. If any

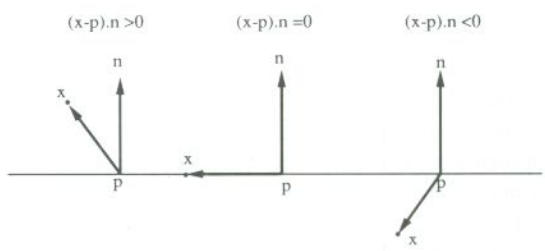

Fig. 2. Collision detection with plane.

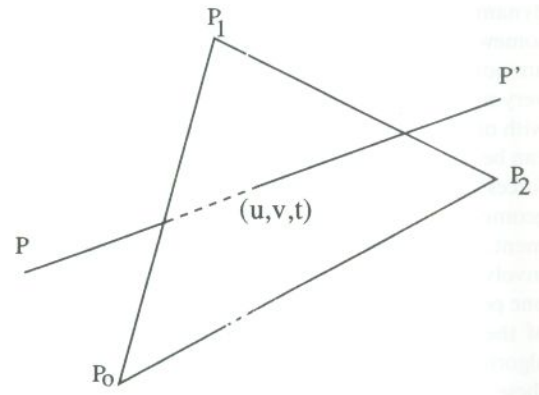

Fig. 3. Path point and triangle intersection.

vertex is on the inward side of all such faces, a collision is detected. The algorithm is developed as the 3D analogue to Cyrus-Beck clipping.

2. Test for penetration of edges of B through the faces of A. Each edge of B is divided into a number of smaller line segments by intersecting it with the infinite planes corresponding to every face of A. The mid-point of each resulting line segment is checked for being inside A by the same method that was used above just for the vertex. Again if any of these mid-points lie inside A the algorithm declares that a collision has taken place.
3. The last test checks for the infrequently occurring case where two identical polyhedra are moving through each other with the faces perfectly aligned. Here the centroid point of each face B is tested against A by the method used for vertices, above. If any of these centroids is inside A the algorithm detects a collision.

In the second phase one reverses the order and compares A against B. If the algorithm survives both the phases without detecting a collision, then the two polyhedra are declared not to penetrate. The algorithm is computationally very expensive and can be speeded up by a variety of tricks like bounding box and bounding sphere.

Baraff[1] takes advantage of the geometric coherence between successive time steps to speed up collision detection test. The algorithm makes use of the fact that a pair of convex objects do not penetrate if and only if a separating plane between them exists. By caching the separating plane as a witness in successive time steps, the algorithm simplifies the decision problem of inter-penetration. Ganter *et al.*[3] use space partitioning, which exploits the spatial coherence technique to detect collision between polygonal faces. The method effectively reduces the computation time due to the fact that the overlap region between the objects at or near impact is usually small compared to the total size of the objects' bounding boxes. As a result most of the faces in the total face set will not be contained within the overlap region and thus will not require testing.

It should be noted that all these algorithms for collision detection would fail if one object completely passes through the other during a single time step. The problem is analogous to aliasing in computer graphics, which primarily arises due to inadequate sampling. The correct solution to this problem is to generalize to 4D space-time swept volume algorithms to detect collision. Moore *et al.*[2] take a more practical approach and either ignores the problem altogether or restrict the animation step size such that it is small compared to the object's size.

Baraff[4] uses a nonpenetration constraint to define a characteristic function ψ. The function ψ intuitively defines a distance between two objects near the point of contact. The function ψ is chosen for each contact point such that it is twice differentiable. The same function is later used to compute the response. Further this scalar valued function is positive, zero, or negative according to whether A and B are disjoint, in contact or inter-penetrating (see Fig. 4).

2.4. *Collision detection for surfaces*

Herzen *et al.*[5] gives a method for collision detection for time dependent parametric surfaces that are continuous and bounded in derivatives. The upper bounds on the parametric derivatives make it possible to guarantee the successful detection of collision and near misses. The method is robust and works with many types of surfaces including bicubic patches.

Baraff[1] uses the concept of extreme distance to formulate the characteristic function for curved surfaces analogous to the one given above for polyhedral objects. However it is easier to define the function ψ for polyhedral objects than for curved surfaces. The extreme distance between A and B near the point of contact is defined as follows. If A and B are disjoint, then the extreme distance between A and B is just the normal minimum distance. If A and B are in contact, then the extreme distance is 0. If A and B have interpenetrated, then the extreme distance is maximum distance between A and B. The extremal points on A and B are the two points P_a and P_b that realize the extremal distance (see Fig. 5).

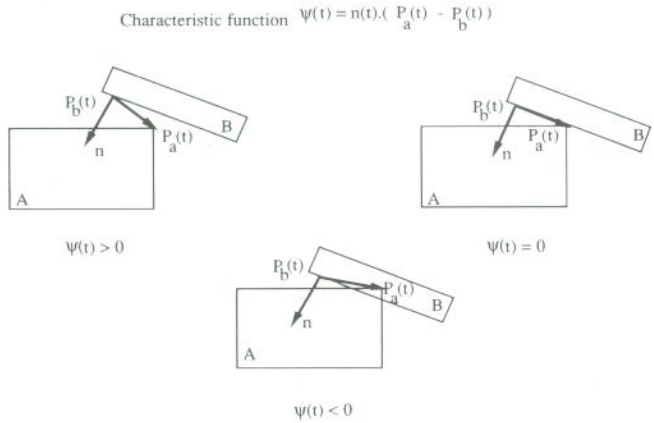

Fig. 4. Characteristic function $\psi(t)$ defining nonpenetration constraint.

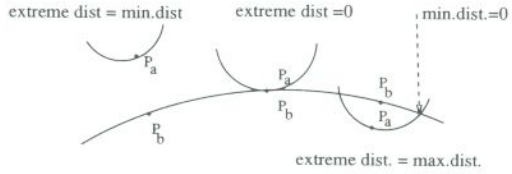

Fig. 5. The external distance and extremal points.

3. COLLISION RESPONSE

Here we shall primarily consider two different approaches to model collision response. Those using analytical methods and those using non-analytical (penalty) methods. In analytical methods there are again two approaches. One using the conservation of momentum principles proposed by Moore *et al.*[2] and Hahn[6], the other based on nonpenetration constraint proposed by Baraff[4]. We shall discuss each method in turn.

3.1. *Collision response using penalty methods*

The penalty method is equivalent to adding a spring to the mechanical system at the point of contact. The advantage of the penalty methods is that they are easy to understand easy to use. They apply equally well to rigid bodies and flexible bodies. However, the main problem with these methods is that they do not fulfill the constraints precisely. Further stiffer springs means stiffer equations that require smaller time steps for accurate numerical integration and hence prove more expensive. Moore *et al.*[2] use this method to simulate response between colliding objects. Thus, when a collision is detected, a very stiff spring is temporarily inserted between the points of closest approach. The spring law is usually k/d, or some other functional form that goes to infinity as the separation d of the two objects approaches 0. k is a spring constant controlling the stiffness of the spring. Lafleur[7] use a slight variant of this technique in cloth animation. They create a very thin force field around the surface to avoid the collision. The force field acts like a shield that prevents the mass points from inter-penetrating.

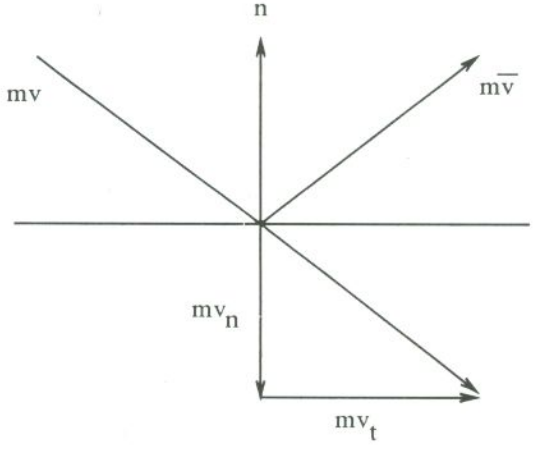

Fig. 6. Reflecting the motion vector of particle.

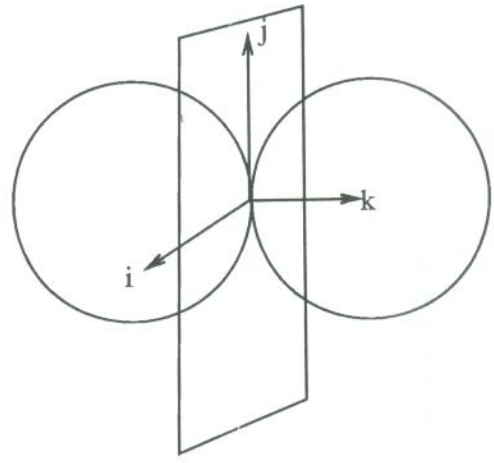

Fig. 7. Particle-particle collision geometry.

3.2. *Collision response using analytical methods*

We shall discuss collision response for different types of object primitives similar to the ones discussed in collision detection.

3.2.1. *Collision response for particles.* Whenever a particle collides with a plane or a surface, it changes its velocity. The new velocity is computed by setting the momentum conservation equation (see Fig. 6).

$$m\bar{v} = mv_t - mv_n$$

Here the momentum vector mv is allowed to rebound against a plane and is decomposed into a normal component mv_n and a tangential component mv_t. This would be a collision without loss of energy. To get a more realistic collision, the particle can be assigned an elasticity parameter η and a friction parameter μ. ($0 < \eta, \mu < 1$). This new motion vector can then be computed as

$$m\bar{v} = (1 - \mu)mv_t - \eta mv_n$$

To simulate particle-particle collision, consider the configuration in Fig. 7.

Here i, j, k represent a coordinate system with i, j defining the collision tangent plane. To get the new velocities of the particle, one needs to solve six linear equations in six unknowns. The first three equations are given by conservation of linear momentum

$$m_1\bar{v}_1 + m_2\bar{v}_2 = m_1v_1 + m_2v_2$$

where

- m_1, m_2 are the masses of the particle.
- v_1, v_2 are velocities of the particles before the collision.
- $\bar{v_1}, \bar{v_2}$ are velocities of the particles after the collision.

The remaining three equations are obtained by assuming the following collision behaviour

$$(\bar{v}_1 - \bar{v}_2)\cdot k = -\epsilon(\bar{v}_1 - \bar{v}_2)\cdot k$$

$$(\bar{v}_1 - v_1)\cdot i = \mu(\bar{v}_1 - v_1)\cdot k$$

$$(\bar{v}_1 - v_1)\cdot j = \eta(\bar{v}_1 - v_1)\cdot k$$

where ϵ is the coefficient of restitution, and μ and η are the coefficients of friction along two perpendicular directions in the collision plane.

3.3. *Collision response for polyhedra*

Moore *et al.*[2] have proposed a solution to the problem based on the conservation of linear and angular momentum. Fifteen linear equations are set up and solved for 15 unknowns. These unknowns are the three components of the resultant linear, angular momentum and the impulse vector. Out of these 15 equations, 12 equations are due to the momentum conservation principle (see Fig. 8):

$$m_1\bar{v}_1 = m_1 v_1 + R$$

$$m_2\bar{v}_2 = m_2 v_2 - R$$

$$I_1\bar{w}_1 = I_1 w_1 + \rho_1 \times R$$

$$I_2\bar{w}_2 = I_2 w_2 - \rho_2 \times R$$

where

- ρ_1, ρ_2 are vectors from the centre of mass of each object to the point of collision.
- m_1, m_2 are the object masses.
- I_1, I_2 are inertia matrices of the objects.
- v_1, v_2 are the linear velocities of objects before the collision.
- $\bar{v}_1$, $\bar{v}_2$ are the linear velocities of objects after the collision.
- w_1, w_2 are the angular velocities of objects before the collision.
- $\bar{w}_1$, $\bar{w}_2$ are the angular velocities of objects after the collision.

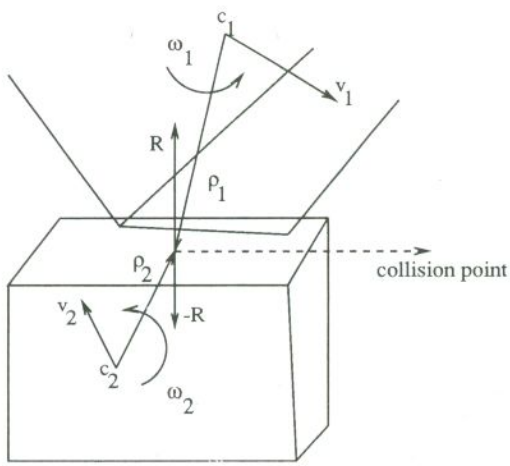

Fig. 8. Two rigid body collision.

- R is the impulse vector, by convention directed from object 2 to 1.

The last three equations depend on the collision behaviour, *i.e.*, elastic or nonelastic collision, with or without friction, *etc*. The square linear system of 15 equations in 15 unknowns is solved by standard Gauss-Jordan or LU decomposition method[8].

If elastic collision is to be modelled, the last three equations are given as:

$$R\cdot i = 0$$

$$R\cdot j = 0$$

$$(\overline{v_2} + \overline{w_2}\times\rho_2 - \overline{v_1} - \overline{w_1}\times\rho_1)\cdot k = -\epsilon(v_2 + w_2\times\rho_2 - v_1 - w_1\times\rho_1)\cdot k$$

where ϵ is the restitution coefficient. In case the collision is non-elastic then $\epsilon = 0$. If friction is to be modelled, a coefficient of friction γ is introduced to express the maximum allowed ratio of impulse parallel versus normal to the tangent plane. The computation is performed in two steps.

First the objects are considered as infinitely rough with no sliding and elasticity. In this case the last three equations of the system of equations become:

$$\overline{v_2} + \overline{w_2}\times\rho_2 - \overline{v_1} - \overline{w_1}\times\rho_1 = 0$$

After solving the linear system, the resulting value of R is examined to ensure that the ratio γ is not exceeded

$$\frac{|R - k(R\cdot k)|}{R\cdot k} < \gamma$$

If the relation is satisfied, the linear system solution gives the new velocities or else the second step must be carried out.

In the second step, the objects are sliding at the point of contact. A limited amount of friction can act against sliding motion. The 3 new equations are

$$R\cdot i = \alpha R\cdot k$$

$$R\cdot j = \beta R\cdot k$$

$$(\overline{v_2} + \overline{w_2}\times\rho_2 - \overline{v_1} - \overline{w_1}\times\rho_1)\cdot k = 0$$

where

$$\alpha = \gamma\frac{R - k(R\cdot k)}{|R\cdot k(R\cdot k)|}\cdot i$$

$$\beta = \gamma\frac{R - k(R\cdot k)}{|R\cdot k(R\cdot k)|}\cdot j$$

As already mentioned earlier[4], models collision as a nonpenetration constraint and gives an analytical method to derive the forces of constraint. Depending upon the relative velocity of approach at the point of contact, the contact is termed as a colliding contact

(hard collision) or a resting contact (soft collision). To simulate response for the colliding bodies, the quantity $\hat{n}(t_0) \cdot v_{rel}^-$ is checked for the sign, where v_{rel}^- is the relative velocity of two colliding bodies just before the collision and $\hat{n}(t_0)$ is normal at the point of contact (see [9] for calculation of normal). If $\hat{n}(t_0) \cdot v_{rel}^- < 0$, then bodies A and B are approaching one another in $\hat{n}(t_0)$ direction and contact impulse J is applied to prevent inter-penetration. Assuming no friction *i.e.*, $J = j\hat{n}(t_0)$ and using the empirical law for collision

$$\hat{n}(t_0) \cdot v_{rel}^+ = -\epsilon \hat{n}(t_0) \cdot v_{rel}^-$$

one solves for j. This method differs from an earlier method suggested by Moore *et al.*[2] where v_{rel}^+, the final velocities are all treated as unknowns. Doing so results in a set of simultaneous equations to be solved. In contrast the Baraff[4] methods regard j as the only unknown quantity and solve a single linear equation in one unknown.

If $\hat{n}(t_0) \cdot v_{rel}^- > 0$ then objects A and B are separating and no contact force need be applied. In case $\hat{n}(t_0) \cdot v_{rel}^- = 0$, objects A and B are in resting contact and it is necessary to apply the contact force in order to prevent inter-penetration. In the absence of friction, the contact force F is written as $F = f\hat{n}$ with f the unknown contact force magnitude at time t_0. Then for each contact point a characteristic function $\psi(t)$ is defined that characterises the geometric distance relation between A and B. Given $\psi(t)$ one can express the constraint that A and B are not penetrating near the contact point by $\psi(t) \geq 0$. Further this constraint is converted into a constraint force by taking the second derivative and constraining the contact force by $\ddot{\psi}(t) \geq 0$

In addition to the geometrically motivated constraint $\ddot{\psi}(t) \geq 0$, there is an additional relationship between $\ddot{\psi}(t_0)$ and f that must be satisfied. If $\ddot{\psi}(t_0) > 0$, then ψ is an increasing function at time t_0 and A and B are separating at the contact point. In this case the contact force is zero. However, if $\ddot{\psi}(t) = 0$ then A and B are not separating and f need not be zero. This complementary condition is written as $f\ddot{\psi}(t_0) = 0$ to express the fact that either f or $\ddot{\psi}$ is zero. Further the restriction that f be non-negative makes the configuration with N contact points satisfy the system of equations

$$\ddot{\psi}_i(t_0) \geq 0, \quad f_i\ddot{\psi}_i(t_0) = 0, \quad f_i \geq 0 \; (1 \leq i \leq N)$$

where f_i and ψ_i are the contact force and constraint function for the ith contact point. For the case of frictionless contact, the above equation forms what is known as a positive semidefinite (PSD) linear complementary problem. It can also be looked as a PSD quadratic programming problem. Baraff[1] advocates its use over his earlier suggested heuristic solution method. However, in the presence of friction, it is known that the above equation is no longer necessarily PSD. Finding the solution of a nonPSD linear complementary problem or quadratic program is NP-hard. Thus, heuristic solution methods may indeed be necessary for practical simulations.

3.4. *Collision response for articulated bodies*

When objects are linked together by mechanical joints, the structure can propagate impulses via the joints (see Fig. 9).

Moore *et al.*[2] have proposed a method based on the solution of a linear system of equations that computes the response for articulated bodies with revolute and sliding joints. The various rigid objects that make up the structure are numbered from 1 to n. For the sake of simplicity let us assume a tree like structure with objects 1 and 2 in colliding contact and the rest linked to one or both of them, either directly or through some number of intermediaries.

For each rigid body we have:

- v_i—linear velocity,
- w_i—angular velocity,
- I_i—Inertia tensor matrix,
- m_i—mass,
- c_i—centre of mass,
- ρ_{ij}—single joint connecting object i to j by vector pointing from c_i to the joint.
- R_{ij}—Attachment impulse, pointing from object j to i ($R_{ij} = -R_{ji}$ if $R_{ij} = (0, 0, 0)$, objects i and j are not connected.)
- R—Collision impulse.

For a collision involving n rigid bodies with $(n - 1)$ joints, *i.e.*, if the objects are all part of one articulated linkage, there are $9n$ unknowns. $6n$ unknowns correspond to resulting linear and angular velocities of the objects, 3 unknowns for collision impulse, and $3(n -$

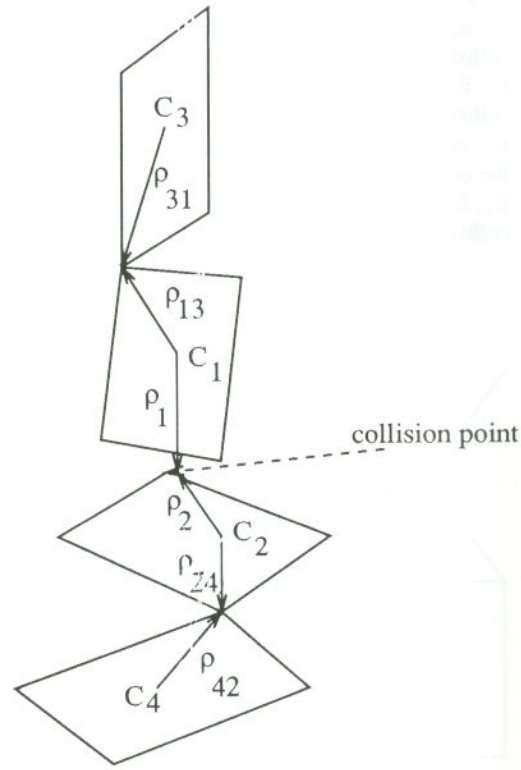

Fig. 9. Articulated body collision.

1) unknowns corresponding to attachment impulses. Thus, the total size of the linear system to be solved is $9n$. The sparsity of the matrix increases as n increases so that if the sparse matrix methods are used the solution should be of the order $O(n)$[2]. The momentum conservation law when applied to objects 1 and 2 give the following 4 vector equations.

$$m_1\overline{v_1} = m_1v_1 + R + \sum_{i=1}^{n} R_{1i}$$

$$m_1\overline{v_2} = m_2v_2 - R + \sum_{i=1}^{n} R_{2i}$$

$$I_1\overline{w_1} = I_1w_1 + \rho_1 \times R + \sum_{i=1}^{n} \rho_{1i} \times R_{1i}$$

$$I_2\overline{w_2} = I_2w_2 - \rho_2 \times R + \sum_{i=1}^{n} \rho_{2i} \times R_{2i}$$

The conditions on the collision impulse R gives rise to 3 scalar equations:

$$R \cdot i = 0$$

$$R \cdot j = 0$$

$$(\overline{v_2} + \overline{w_2} \times \rho_2 - \overline{v_1} - \overline{w_1} \times \rho_1) \cdot k = 0$$

For objects that are not directly colliding (for objects $i = 3 \cdot \cdot n$), the momentum conservation gives rise to $(n - 2)$ vector equations:

$$m_i\overline{v_i} = m_iv_i + \sum_{j=i}^{n} R_{ij}$$

$$I_i\overline{w_i} = I_iw_i + \sum_{j=1}^{n} \rho_{ij} \times R_{ij}$$

Each revolute joint connecting objects i and j gives one more constraint vector equation:

$$\overline{v_i} + \overline{w_i} \times \rho_{ij} = \overline{v_j} + \overline{w_j} \times \rho_{ji}$$

The method can be extended to articulated bodies with loops and structures with sliding joints.

4. CONCLUSIONS

Collision detection and response is one of the most difficult of behaviours to handle in a dynamic simulation. Traditionally, collision detection and response were treated as two different problems. One reason for this is that the underlying disciplines are not the same. Typically, geometry and dynamics do not have much in common. The object representation in each of these disciplines is different. For example, the geometric modelling systems try and represent the shape and the associated shape operations on the object, where as dynamic analysis systems represent the object with mass, inertia, and the associated equations of motion. However, in physically based systems, an attempt is made to integrate these[10]. The early work used computational geometry algorithms to solve the collision detection problem and the law of conservation principles to solve the response problem. As a result the early systems were computationally prohibitive. As an immediate remedy to the problem gross approximations were done to improve the performance. It is important for one to realize that in physically based systems, it is necessary to treat collision detection and response as one problem and deal with it within a single framework. At present, systems based on nonpenetration constraint take such an unified view but have some problems in choosing the correct characteristic function and the inability to give a good computational model to represent friction.

Acknowledgements—This work was carried out at the Graphics & CAD Division of the National Centre for Software Technology (NCST), Bombay during my study leave period from the University. I am grateful to Dr. S. P. Mudur, for his continued intellectual, emotional, and technical support. I thank Dr. S. Ramani, Director NCST for providing a free research atmosphere and the Goa University authorities for their support without which this work would not have been possible.

REFERENCES

1. D. Baraff, Curved surfaces and coherence for non-penetrating rigid body simulation, *Comp. Graph.* **24,** 19–28 (1990).
2. M. Moore and J. Wilhelms, Collision detection and response for computer animation. *Comp. Graph.* **22,** 289–298 (1988).
3. M. A. Ganter and B. P. Isarankura, Dynamic collision detection using space partitioning. *J. Mech. Des.* **115,** 150–155 (1991, July).
4. D. Baraff, Analytical methods for dynamic simulation of non-penetrating rigid bodies, *Comp. Graph.* **23,** 223–231 (1989).
5. B. Von Herzen, A. H. Barr, and H. R. Zatz, Geometric collision for time-dependent parametric surfaces. *Comp. Graph.* **24,** 39–48 (1990).
6. J. K. Hahn, Realistic animation of rigid bodies. *Comp. Graph.* **22,** 299–308 (1988).
7. B. Lafleur, M. N. Thalmann, and D. Thalmann, Cloth animation with self-collision detection, *Animation of Synthetic Actors and 3D Interaction.* Thalmann M. N. and Thalmann D. November 1991.
8. G. Hégron and B. Arnaldi, *Computer Animation: Motion and deformation control*. Eurographics Tutorial (1992).
9. D. Baraff, Non-penetration constraints *ACM SIGGRAPH Course Notes* No. 23. An introduction to physically based modeling, H30-H48 (1991).
10. J. F. Cremer, *An Architecture for General Purpose Physical System Simulation—Integrating Geometry, Dynamics, and Control*. Doctoral Thesis, Cornell University, Department of Computer Science, TR 89-987. (1989).