Purdue University
Purdue e-Pubs

Department of Computer Science Technical Reports

Department of Computer Science

1991

Collision Detection and Analysis in a Physically Based Simulation

William J. Bouma

George Vaněček

Report Number:
91-055

Bouma, William J. and Vaněček, George, "Collision Detection and Analysis in a Physically Based Simulation" (1991). *Department of Computer Science Technical Reports.* Paper 895.
https://docs.lib.purdue.edu/cstech/895

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

COLLISION DTECTION AND ANALYSIS IN A PHYSICALLY BASED SIMULATION

William J. Bouma
George Vanecek, Jr.

CSD-TR-91-055
July 1991

Collision Detection and Analysis in a Physically Based Simulation*

William J. Bouma and George Vaněček Jr.

Computer Science Department
Purdue University
West Lafayette, IN 47907, U.S.A.

Abstract

We consider the geometric support in detecting and analyzing collisions and contact between arbitrarily shaped polyhedral objects for a physically based simulation. The contact detection is formulated as a static collision-detection problem in three-dimensional space. We address both robustness and efficiency of the problem, and show how both can be achieved by using the brep-index data structure.

1 Introduction

A computer simulation of physical systems that is based on rigid-body dynamics and that involves objects with arbitrary shapes requires the services of a geometric modeling system. The geometric modeling system is used initially to establish mass properties and to formulate constraints for the system's dynamics. When simulating the motion of objects in the presence of obstacles with possible collisions and prolonged contact, the geometric modeling system is used throughout the simulation. The dynamics modeler must inquire at each time step whether two bodies are about to collide, and if so, obtain a geometric analysis in the vicinity of the contact points. The results of the geometric analysis are then used to formulate a system of equations for evaluating the dynamical consequence of the collision. As an example of the type of simulations being described, Figure 1 shows eight frames of a simulation in which a double-linked chain with ten rings tumbles when the second top ring is held fixed [11].

In order for the simulator to formulate the correct system of equations for each time step, it must distinguish between *collision contacts* which exist for an infinitesimal amount of time and *temporary contacts* which persist for a measurable

*To appear in the Second Eurographics Workshop on Animation and Simulation, Sept 1-2, 1991, Vienna Austria.

duration. As an example, Figure 2 shows two rings in temporary contact with the top ring fixed in space and the bottom ring sliding—the contact points and normals are indicated by the vectors. In the case of collision, the generated equations describe an instantaneous velocity change between the impacting objects. In the case of temporary contact, new equations are added to the existing set which constrain the relative motion of the objects at the contacting points. For the geometric system to compute contact normals meaningful to the dynamics, it is not sufficient to use only the topology local to each contact point. As long as the dimension of the contacting regions do not change, the normals should not vary significantly between subsequent time steps of the simulation. This kind of reasoning can only be done if the geometric system is informed at each time step about the temporary contacts currently in effect.

For practical considerations, both the geometric and the dynamics systems use finite-arithmetic computations. Consequently, inaccuracies occur in the simulations. In the geometric system, exact contact cannot be determined. In the dynamics system, perturbations appear in solving the equations of motion. These numerical errors may lead to a difference in the states of the two systems. Methods of avoiding or undoing such inconsistent states must be provided. For example, the dynamics system may consider two objects in contact while the geometric system detects and reports the objects interpenetrating. The geometric system, knowing where the temporary contacts occur, thus needs to tolerate such penetrations. The dynamics system may use the report of penetration at a constrained contact to apply a force to correct the penetrating point back to the object's surface. An interpenetration at a point where there is no temporary contact occurs as a result of using too large a time interval in the simulation loop. In this case, the geometric modeler must determine the depth of interpenetration and the previous time step must be recomputed with a smaller time step to determine the contact within a given tolerance of the object's surface.

We consider the use of model-driven rigid-body dynamics simulators, such as Newton [9], that have an event handling mechanism for dealing with discontinuities in the simulation, and which use a geometric modeling system as a server. For such simulators, the contact detection and analysis can be solved as a static problem at each time step. From a geometrical point of view, the contact detection and analysis problem can be reduced to a line/solid classification problem for which there is an efficient and robust solution. The use of complex geometrical shapes and the massive number of contact queries and analyses suggests using specialized solid representations that facilitate and optimize such geometric computations.

In this paper, we describe the interface between the Newton dynamics simulation system [17, 18] and Protosolid [27], a geometric modeling system. This work extends the system introduced in [28]. We are mainly concerned here with analyzing the geometry in a meaningful way for the dynamics. Enough of the

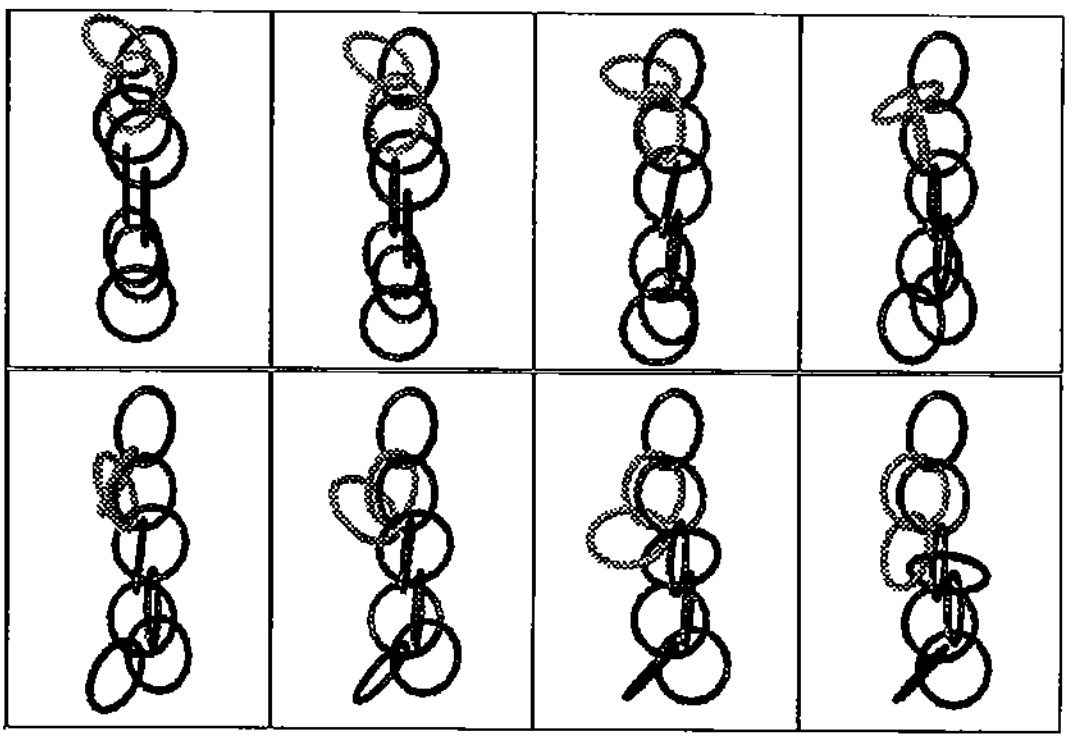

Figure 1: Eight frames of the tumbling rings.

Figure 2: Two rings in temporary contact with the contact points and normals indicated by the vectors. The top ring is fixed while the bottom ring is free to move.

dynamics are presented to show why we handle the geometry a certain way, but the details of handling the dynamics of temporary contact will be discussed in a forthcoming paper. We begin this paper by reviewing the related work in applying geometry to dynamics simulations. In Section 3 we outline the dynamics engine of the Newton system. We state our problem precisely in Section 4, and in Sections 5 and 6 we propose a model of contact and analyze the geometric issues in temporary contact. Handling exceptional events is discussed in Section 7. We show in Section 8 that temporal coherence is necessary to get the normals right. Once contact is understood, we give the interface between the dynamics module and the geometric module in Section 9. In Section 10 the geometric contact analysis performed by the geometric module is outlined. The work presented in this paper is then discussed in the last section.

2 Related Work

The collision detection problem is formulated either as a static checking for interpenetration given the positions and orientation at a given time or as a dynamic problem over a given time interval given the motions of the objects as functions of time.

There are many different algorithms for detecting or avoiding collisions of convex objects. Chazelle and Dobkin [7] show that when two convex polyhedra with p and q faces are given in a suitable representation, their penetration can

be determined in $O(\log^3 n)$ time, where $n = p + q$. Since computing the intersection of two convex polyhedra requires $O(n \log n)$ time [23], it is computationally less expensive to find only whether two convex objects intersect (i.e., collide or interpenetrate) rather than to compute the entire nonregularized intersection.

Gilbert, Johnson and Keerthi show a linear time algorithm for computing the shortest distance between two nonintersecting convex objects [12, 13]. When the objects interpenetrate, Cameron and Culley [5] further provide a measure of the depth of interpenetration to yield information on how deeply the two objects dug in. The efficient evaluations of the distance functions are useful in motion planning and collision avoidance problems. An alternate approach to collision free motion planning problem is characterized by Lozano-Pérez and Wesley [16]. They consider fixed objects around which a single moving object has to navigate. An unobstructed path can be obtained by considering the moving object as a single point and by considering each fixed object as being enclosed in a forbidden region of sufficient size. As long as the point does not penetrate a forbidden region a collision does not occur.

The methods presented above are based on static collision detection. Several researchers have investigated the problem based on the object's configuration space. For prespecified trajectories consisting of a sequence of individual translations and rotations about an arbitrary axis, Boyse [3] gives an algorithm for detecting and analyzing collisions between a moving and a stationary object. His algorithm works for nonconvex objects represented as B-reps. Canny [6] gives an algorithm for computing the exact points of collision for objects that are simultaneously translating and rotating. It can deal with any path in configuration space that can be expressed as a polynomial function of time. The objects are assumed to be a union of overlapping convex objects.

Obtaining a local geometrical analysis of simple collisions was presented by Hahn [14], and Moore and Wilhelms [21].

To guarantee that no collisions are missed when using the static analysis approach, several researchers have investigated the four-dimensional time-space problem [22, 5].

Recently, Baraff [2] showed how to analyze collisions for objects with curved-surfaces, and take geometric coherence into account.

3 Background

Most of the previous research has focused on correctly formulating collision forces, and in dealing with the dynamics given a simplified geometric analysis of the contacts. Relatively little work has been done in fully analyzing the geometric contact of complex objects within physically-based simulation systems. To understand the problem from the dynamics point of view we begin by briefly reviewing the dynamics engine of the Newton simulation system. For full detail we refer the

reader to Cremer [8].

With a high-level definition language, an initial configuration is given which defines primitive and composite objects, their positions and orientations, various constraints, and local and global properties. The specification of the initial configuration is parsed and yields a set of ordinary differential equations of motion, one for each object. These are based on the Newton-Euler equations. They are

$$\begin{aligned} m\ddot{r} &= F \\ J\dot{\omega} + \omega \times J\omega &= T, \end{aligned}$$

where m is the mass, $\ddot{r}$ is the acceleration of the center of mass, J is the 3-by-3 inertia matrix, F and T are the force and torque on the object, and ω and $\dot{\omega}$ are the angular velocity and acceleration. Optionally, the motion of two objects may be constrained in relation to each other by the introduction of various hinges, such as the ball and socket hinge, and the pin hinge. Each hinge specification is given as a set of positional constraint equations. These equations are differentiated twice to yield an acceleration constraint which is then added to the set of motion equations. For example, a ball and socket hinge, imposes the constraint that a specific point on the first object remains in contact with a specific point on the second. The corresponding equation is

$$r_i + c_i = r_j + c_j,$$

where r_i is the position of the ith object's center of mass, and c_i is the vector from the center of mass to the location of the hinge point. The second derivative yields the motion equation

$$\ddot{r}_i + \dot{\omega}_i \times c_i + \omega_i \times (\omega_i \times c_i) = \ddot{r}_j + \dot{\omega}_j \times c_j + \omega_j \times (\omega_j \times c_j).$$

The addition of the hinge equation results in an unknown force between the two objects at the contact point. The force X is thus added to the motion equations for object i and subtracted from those of object j. The new equations for object i become

$$\begin{aligned} m_i\ddot{r}_i &= F_i + X \\ J_i\dot{\omega}_i + \omega_i \times J_i\omega_i &= T_i + c_i \times X. \end{aligned}$$

Newton has a wide variety of hinges to restrict the various degrees of freedom between objects. Hinge equations can be added to or subtracted from the set of motion equations while a simulation is in progress. We call such events *creating* and *breaking* hinges.

During a simulation an object in motion may come into contact with some other object. At a contact point ρ, a nonnegative contact velocity in the normal direction n indicates that the objects are either in contact or are separating; a

negative contact velocity indicates that a collision is taking place. The surface normal is determined from the geometry as described in Section 10. The velocity of a point $\rho = r + c$ on an object is given by

$$\dot{\rho} = \dot{r} + \omega \times c.$$

The relative velocity of the contact points in the contact normal direction is

$$(\dot{\rho}_1 - \dot{\rho}_2) \cdot n.$$

Newton models rigid body impact as an instantaneous change in velocity. The velocity of a colliding point before impact is written $\dot{\rho}^B$, and after $\dot{\rho}^A$. Because we are dealing here with frictionless systems, each scalar impulse force, f_i, occurs in the contact normal direction. For one object the impact equations are

$$\begin{aligned} m \cdot (\dot{r}^A - \dot{r}^B) &= f_1 n_1 + f_2 n_2 + \ldots + f_n n_n \\ J \cdot (\omega^A - \omega^B) &= f_1 \rho_1 \times n_1 + \ldots + f_n \rho_n \times n_n. \end{aligned}$$

Each collision also contributes to the system of equations one scalar equation of the form

$$(\dot{\rho_i}^A - \dot{\rho_j}^A) \cdot n = -e(\dot{\rho_i}^B - \dot{\rho_j}^B) \cdot n.$$

Here $\dot{\rho_j}^B$ is the velocity of the collision point on object j before impact, and e is the coefficient of restitution, which allows for kinetic energy loss in the impact. Note that hinges transmit impulsive forces during impact. Equations and terms are added to the set of impact equations in a manner similar to that of adding a hinge to the basic motion equations.

Nonimpact contact occurs when the contact velocity in the contact normal direction is zero. Newton models this situation by creating a hinge between the objects which describes the geometry of the contact. For instance a vertex of one object touching a face of the other causes the creation of a point-on-plane hinge constraint. This constraint allows the vertex to slide freely within the face, but does not allow it to penetrate the solid. The acceleration constraint (assuming the plane is on object j) is

$$(\ddot{\rho}_i - \ddot{\rho}_j) \cdot n + 2(\dot{\rho}_i - \dot{\rho}_j) \cdot (\omega_j \times n) = 0.$$

This equation should actually be an inequality since the point must be allowed to leave the plane in a direction away from the contacting object. Instead, Newton models inequality constraints by combining equality constraints with event handling. For a hinge with an inequality constraint, if any noncompressive force is detected at the hinge point, the hinge must break. The hinge could also be broken by the vertex geometrically leaving the face (e.g. Figure 5(b)). We call such hinges that come and go during the simulation *temporary hinges*. Occurrences of new contact or changes in temporary contact are known as *exceptional events* because they cause discontinuities in the simulation.

The simulation loop iterates time. At each time t, a complete and consistent state is obtained which consists of all the objects' positions, orientations, velocities and accelerations as well as the current temporary hinges. The current state is saved and a new state for time $t + \Delta t$ is computed, for a prespecified time interval Δt. The simulator then checks if any exceptional events have occurred during the Δt interval. If an event is detected, the system isolates the time of the event occurrence to within a specified tolerance, computes the state of the complete system at that time, and effects any changes to the state caused by the event. The iteration proceeds with t set to the time of the currently established state.

4 Problem Assumptions

To detect and handle the exceptional events we have to check the spatial interactions of two arbitrarily shaped objects very frequently. To do this, we make several assumptions.

The first assumption is that the objects are rigid-bodies. The second assumption is that the objects are polyhedral with arbitrary genus. The use of rigid bodies with planar faces simplifies and improves the efficiency of the geometric module. The objects are assumed to be given as boundary representations (B-reps). A preprocessing step attaches a multi-dimensional partitioning tree structure to each B-rep to serve as a volumetric index into the B-rep. This combined structure is called the *b-rep index* [26]. Its use greatly improves the efficiency of the collision detection and analysis algorithm.

The third assumption is that there is no friction. A direct consequence of this assumption is that each contact region can be modeled by a finite number of points of the convex hull.

The fourth assumption is that the relative motion of an object during the interval Δt is sufficiently small so that the collision of two impacting objects is detected. Without this assumption, the space-time sweep of all objects would need to be modeled in four-dimensional space to guarantee that all collisions are detected.

5 Describing Contact

We are given two nonpenetrating objects that are in contact. Since the objects need not be convex, the area of contact can be geometrically quite complex. Fortunately, it is sufficient to describe the intersection by a finite number of point contacts. For example, the points along a face/edge contact may be modeled by the endpoints of the edge; the points in a face/face contact may be similarly modeled by the vertex points of the enclosing convex hull of the face/face in-

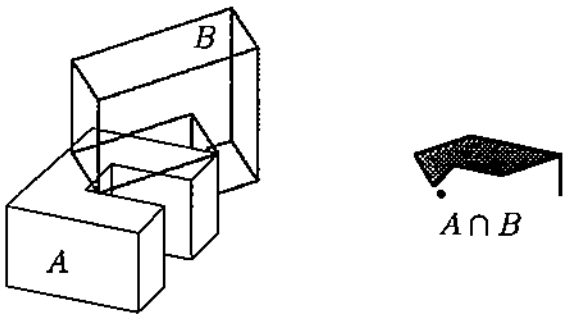

Figure 3: Two touching objects and their nonregularized intersection.

tersection [10]. Here x/y indicates the topological entity x of solid s_i in contact with topological entity y of solid s_j, where the topological entities are the vertices, edges and faces.

Definition 1 *Let s_i and s_j be two solids that touch but do not penetrate each other's interior. Their intersection can be described in terms of the intersecting topological entities by*

$$\mathcal{I}_{ij} = \{x/y \mid x \text{ is an entity of } s_i, \; y \text{ is an entity of } s_j, \text{ and } x \cap y \neq \emptyset\}.$$

Consider the two objects of Figure 3 and the set $\mathcal{I}_{ij}$ consisting of 16 pairs of entities in contact. We see that all except three of the pairs form k-dimensional regions bordering other $(k+1)$-dimensional regions. The three regions do not bound any higher-dimensional regions. It is these regions that we are interested in. By organizing the intersection in terms of these regions, we can identify which regions constitute temporary contact and which constitute collisions. Furthermore, for each region resulting from temporary contact we need only to compute a single contact normal. For example, the set-theoretic intersection of the two solids in the figure consists of three contact regions; a 0D vertex/edge, a 1D edge/edge, and a 2D face/face.

Definition 2 *The set of contact regions is $\mathcal{R}_{ij} \subset \mathcal{I}_{ij}$ such that for every unique pair R_1 and R_2 in $\mathcal{R}_{ij}$,*

$$R_1 \cap R_2 = \emptyset \text{ or } \dim(R_1 \cap R_2) < \min(\dim(R_1), \dim(R_2)),$$

where $\dim(R)$ is the dimension of the support space of R.

A region is homogeneously either a zero, one or two dimensional. That is, a 0D region is coincident with a point, a 1D region is contained in a line, and a 2D

regions is contained in a plane. A 0D region can result from either vertex/vertex, vertex/face, face/vertex, vertex/edge, edge/vertex, or transversal edge/edge contacts. A 1D region can result from either edge/face, face/edge or from collinear and overlapping edge/edge contacts. A 2D region can only result from a face/face contact. We treat the regions of $\mathcal{R}_{ij}$ as closed sets so that a region contains its boundary. For example, a face/face contact region includes all the bordering vertices and edges. This also means that two adjacent regions may share a boundary line or a boundary point. Thus, although each region has a single contact normal or impulse, a single point may have several normals and impulses. A region may also be disconnected. For instance, a face/face contact region may consist of several disjoint components.

6 Temporary Contact

The situation arises that objects may be in contact but not be colliding. For a given contact region this occurs when the relative velocities at the points of contact lie within the plane of contact. In order to keep two objects apart at the various contact regions, the dynamics module creates a temporary hinge between the objects. The hinge adds kinematic constraints to hold the objects from interpenetrating.

A temporary hinge over a set of contact regions is kept for as long as there is a compressive force at some point in one of the regions and the relative velocity at that point is in the contact plane of that region. When no such point exists for any region of the temporary hinge, it is removed, since this means that the objects are, or will be, moving apart.

Since we are dealing with polyhedral objects the types of contact are limited to 0D, 1D, or 2D regions on a plane, 0D or 1D regions on an line, or 0D regions on points. The nD region on plane contact is modeled by $n + 1$ point on plane constraints. The edge contact is modeled differently according to the geometry. If the 0D contact is a result of two edges crossing it is modeled as an edge-edge constraint, else we model it using point-plane. The difference between the two constraints is that the normal is given by the contact plane in the point-plane case, and it is given by the cross product of the two edges in the edge-edge case.

Each contact is described as a point on object s_i and a point on object s_j which lie on their boundaries, so the two points need not necessarily be coincident in global object space. Contact is detected when the object boundaries are within a small prespecified distance of each other. Temporary contacts, however, are allowed to interpenetrate by a distance in excess of the collision tolerance. This is necessary, as the integration of the motion equations results in small positional drifts. Refer to Figure 4. When the dynamics module declares a temporary hinge over a given contact region, the geometric module ignores any penetrations occuring at that region and returns each of its contacts c as two separate points

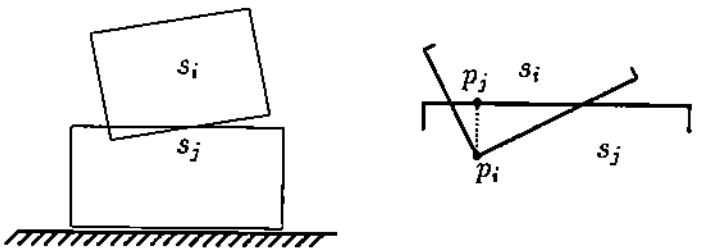

Figure 4: Allowed positional drift during the temporary contact of two objects.

p_{ci} and p_{cj}, one on s_i and one on s_j. To prevent the points from penetrating too deeply due to numerical error, a correction force is added to pull the penetrating point towards the contact plane.

$$\ddot{n} \cdot p_{\text{rel}} + K_v(\dot{n} \cdot p_{\text{rel}} + n \cdot \dot{p}_{\text{rel}}) + K_p n \cdot p_{\text{rel}},$$

where $p_{\text{rel}} = \rho_1 - \rho_2$ is the relative positional vector, $\dot{p}_{\text{rel}} = \dot{p}_{\rho_1} - \dot{p}_{\rho_2}$ is the relative velocity and K_v and K_p are velocity and positional constants.

Allowing the temporary contact points to drift will prevent the simulation from breaking due to inconsistencies in the current contacts. But it can lead to problems in the detection and handling of future contact changes. Consider the following situations:

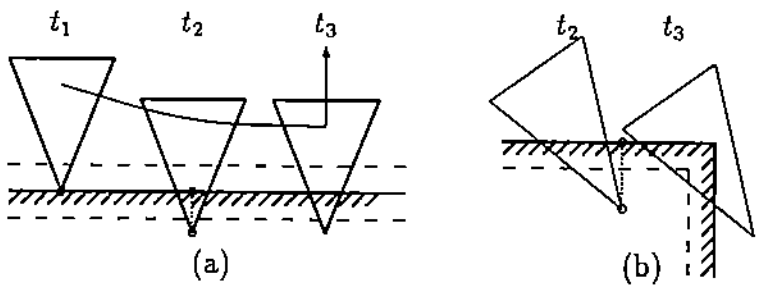

Figure 5: Examples of breaking a temporary hinge while the objects interpenetrate due to (a) dynamics and (b) geometric changes.

Scenario 1. Two objects touch and remain in contact as shown in Figure 5(a). The dynamics module establishes a temporary hinge. (i.e., time t_1). The contacts are reported to the geometric module which during the life of the temporary contact allows small interpenetrations (i.e., time t_2). The dynamic module corrects the interpenetrations as necessary. Now consider the result if the object

on top gains an upward velocity and begins to separate from the object on the bottom (i.e., time t_3 of (a)), or it simply slides off an edge (i.e., (b)). Since at this time the two objects are interpenetrating by more than the collision tolerance (as indicated by the dashed lines), the breaking of the temporary contact causes an interpenetration to be detected and reported. Attempting to resolve the interpenetration, the dynamics system searches for the time of first contact in the interval between t_2 and t_3. Since the interpenetration already existed at t_2, no such initial contact can be found.

Scenario 2. Object B is sliding across Object A, as shown in Figure 6(a). The two temporary contacts are indicated by circles. In Figure 6(b), the right contact point on B interpenetrates Object A. This time, the interpenetration is not the result of allowed drift in a contact. Rather, a new geometric entity has been encountered by a previously established temporary contact.

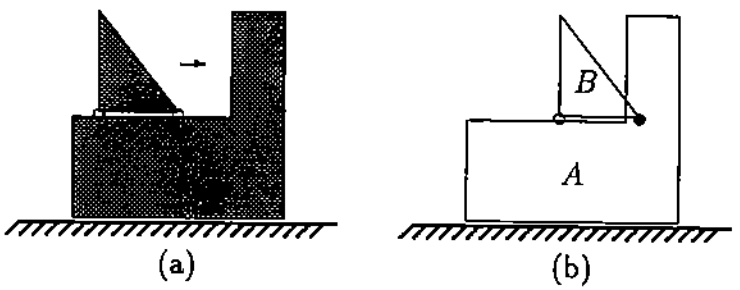

Figure 6: An example where a temporary hinge of Object B becomes a collision.

7 Exceptional Events

Accurate detection and handling of contact change events is essential for realistic simulation. To this end, the dynamics module maintains a list of the temporary hinges in effect at the last time step. The contact points for the regions over which a temporary hinge applies are separated into three categories:

Active These are the support contacts currently associated with the temporary hinge. They are obtained initially by solving a quadratic programming problem over all the contact points of the regions associated with the hinge [10, 20]. All active contacts exert compressive forces in the contact-normal direction.

Inactive These are the remaining zero-velocity contacts which are not used in formulating the temporary hinge equations. The active and inactive contacts may change during the life of a temporary hinge.

Dead These are the contacts which have positive velocity in the normal direction. These cannot be used in formulating the hinge equations. However, they still persist due to geometric contact or interpenetration.

An exceptional event is indication that the state of some currently known contact point has changed, or a new contact has occured. There are two types of events which will be detected by the geometric modeler, and two that are found by the dynamics module. For details on how the geometric events are detected, see Section 9. The contact events are:

1. An active contact point has left its known support
2. A new contact point has been found.
3. A negative velocity in the normal direction is found at a contact point (see Figure 5(a)). (This occurs as a result of some external force such as a collision.)
4. An active contact point obtains a positive acceleration.

Events must be handled in the order in which they occur. Thus, upon discovery that there has been an event during a time step the simulator goes into a loop to isolate the set of events which occur first. The dynamics modeler then resolves the events in the following order:

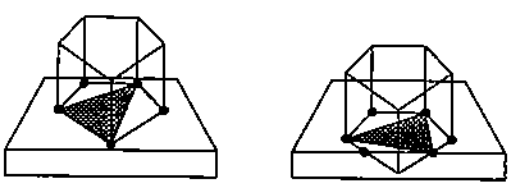

Figure 7: An active contact point breaks and a new support region (shown shaded) is created.

1. The way Type 1 contacts are handled depends on the geometry. If other live contacts remain for the hinge region the support changes to a different set of contacts (see Figure 7) or the hinge breaks completely. In the later case if a penetration persists as a remnant of the old hinge drifting, a new contact point is created between different entities (see Figure 6(b)).
2. The new points are added to the set of all contacts.
3. While there exist negative velocity contacts, formulate and solve collision equations.

4. The contacts with positive normal acceleration become inactive.

After performing these steps, all contacts have normal velocities which are either positive or zero. Any zero velocity contacts which belong to an existing hinge are added to the inactive contact set of the hinge. Any positive velocity contacts which belong to an existing hinge are moved from the inactive to the dead contact set. New hinges are created for any zero velocity contacts which do not find a match to an existing hinge. For each temporary hinge, the new set of active contacts are found, and the equations re-established.

8 Temporal Coherence

Consider a block sliding at time t as shown in Figure 8. Each vertex/face contact on the bottom face of the block has an associated contact normal, namely the perpendicular to the plane of the face, n. At time $t + \Delta t$, an edge of the block comes into contact with one of the supports. The 0D region is a vertex/edge contact which according to the formulation given in Section 10 has a different normal from that of the point/plane. In the example the new normal turns out to be in the exact opposite direction to the velocity of the block. Thus there will be a collision at the point causing the block to bounce in an unnatural manner.

In general, this problem will occur whenever a contact normal n for a sliding object is chosen such that $n\dot{v} < 0$, where v is the relative velocity vector at the contact point of that normal. Since the point on edge normal is really indeterminate, in this case, it is beneficial to choose the normal such that $n\dot{v} >= 0$. However, this clearly would not be the correct normal to choose in all cases, (for example if there really was impact at the point). There is no purely geometric formulation for vertex/edge or vertex/vertex that will provide normals that give natural behavior in all situations. The formulas in Section 10 provide very realistic behavior in the case of impact. For temporary contact we rely instead on coherence between time steps.

In the state just prior to the vertex/edge contact, the contact was vertex/face, and n_1 was perpendicular to the contact velocity. Since the normal of the vertex/edge is indeterminate, we can just let it be the same as it was in the previous state. How we determine that this new vertex/edge contact actually corresponds to the vertex/face contact in the previous time step will be explained in the following sections. Given that this can be done, the problem is solved, if there is previous state information. If the simulation begins with objects having vertex/edge contacts, so that we must rely solely on the geometric formulas to compute the normals, those objects may exhibit strange behavior.

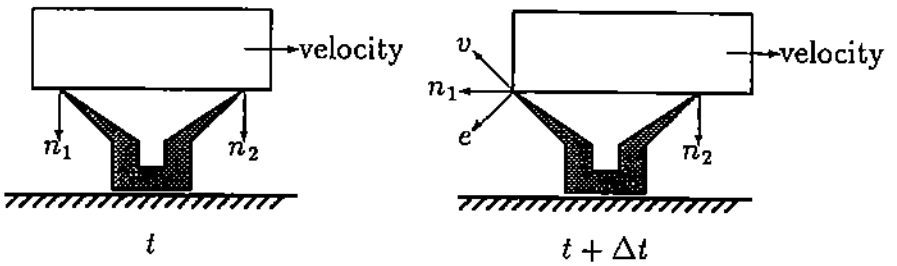

Figure 8: Side view of a block sliding across a four finger support.

9 Information Exchange

Consider the consistent state at time t. The dynamics module has for each object s_i its position, r_i, the orientation, q_i, the velocity $\dot{r}_i$, and the angular velocity ω_i, along with the various properties and constraints. Geometrically, some of these objects may be instances of the same solid. That is, for a brick wall the geometric module needs to model only one solid representing a brick. Nevertheless, the dynamics module knows nothing about the objects' boundaries or extents. This information is kept by the geometric module which maintains a unique set of solids each of which may be referenced by several objects. In addition, it maintains a table, $\mathcal{T}$, that keeps information about the current temporary contact regions:

$$\mathcal{T}_{ij} = \begin{cases} (s_i, s_j, M_i, M_j, \mathcal{C}_{ij}) & \text{if } i < j \\ \mathcal{T}_{ji} & \text{if } j > i \\ \text{undefined} & \text{if } i = j \end{cases}$$

for $1 \leq i, j \leq m$ where m is the number of objects, $\mathcal{C}_{ij} \subseteq \mathcal{R}_{ij}$ is the set of temporary contact regions, and M_i and M_j are transformation matrices as described below. The objects are created so that their centers of mass are at the origin of their corresponding local coordinate spaces.

Now consider time $t + \Delta t$. The dynamics module computes a tentative state for time $t + \Delta t$ and gives the new transformational matrices $M_1, \ldots, M_n$ for the objects to the geometric module. Each M_i is a 4x4 transformation matrix that maps solid s_i from its local frame to the global reference frame. Given the transformation matrices, the geometric module then checks for collisions and contacts. If any two objects interpenetrate, further geometric processing for time $t+\Delta t$ ceases and the interpenetration is reported to the dynamics module. When no interpenetration due to missed collisions is detected, the collision and contact regions, $\mathcal{R}$, are determined and given to the dynamics module. For each pair of touching objects s_i and s_j, the regions in $\mathcal{R}_{ij}$ are given in the following form. For

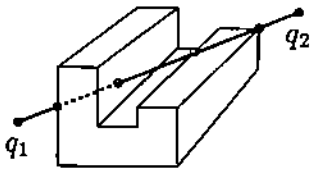

Figure 9: Classifying edge e with endpoints $[q_1, q_2]$ with respect to a solid.

contact regions lying in a plane, the reported geometry is

$$[\overline{n}_j, m, [l_1, p_{i1}, p_{j1}], \ldots, [l_m, p_{im}, p_{jm}]],$$

where $\overline{n}_j$ is the contact normal in the local space of s_j, m is the number of contact points reported, and for each contact point, p_{ik} is the point of contact on s_i, p_{jk} is the point of contact on s_j, and l_k is a unique contact point label. During the life of a given contact region, each unique contact point maintains a unique label to differentiate it from other contact points in the region. This is necessary in recognizing the exceptional events of an active contact point breaking or a new contact point appearing.

For transversal edge/edge contact regions, the reported geometry is

$$[\overline{e}_i, \overline{e}_j, [p_i, p_j]],$$

where $\overline{e}$ is the edge direction.

For each reported $\mathcal{R}_{ij}$, the dynamics module partitions the contact regions into the set of colliding regions and the set of temporary contact regions, C_{ij}. The temporary contacts are then identified to the geometric module which retains them for analysis during the next time step.

10 Geometric Contact Analysis

In this section we describe the geometric analysis of two objects s_i and s_j by the geometric module. Before any analysis begins, the intersection of the spherical extents of the two objects is checked. If the extents do not intersect, the objects do not intersect either and further analysis is unnecessary. If they do intersect, then full analysis follows.

First we compute the mapping matrices M_{ij} and M_{ji}, where $M_{kl} = M_l^{-1} \cdot M_k$. These enable us to map the boundary of s_k into the local space of object s_l, so that we may easily classify the boundary of s_k against the volume of s_l. For reasons of efficiency and robustness [19], the boundary classification is based only on the classification of edges. We classify all the edges of $M_{ij}s_i$ against s_j, and all

the edges of $M_{ji}s_j$ against s_i. With each edge, we associate the edge classification so that after all the edges are classified we can analyze the contacts and resolve interpenetrations.

Consider classifying edge e of Figure 9 in relation to the shown solid. This can be done both efficiently and robustly using the Brep-index data structure—a multidimensional spatial partitioning tree added as an index to a boundary representation [19]. Given that the edge spans the closed interval between points q_1 and q_2, the *classification* of the edge, cl(e), from left to right as shown in the figure is

$$[(q_1, \mathbf{out}), \mathbf{out}, (e_1, p_1), \mathbf{in}, (f_1, p_2), \mathbf{out}, (e_2, p_3), f_2, (v_1, p_4), \mathbf{out}, (q_2, \mathbf{out})],$$

with the pair (x,p) corresponding to the edge's penetration of entity x at point p. The edge classification tells us not only whether the edge is inside, outside or on the boundary, it tells us where the other solid is penetrated and through what entity.

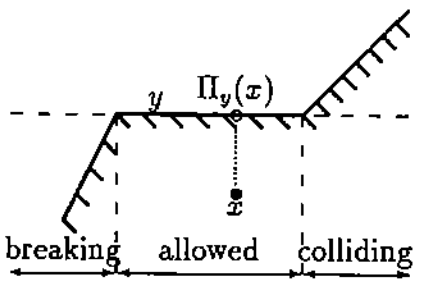

Figure 10: Allowed interpenetration for a temporary contact x/y.

After all the edges of both objects are classified, we identify all the interpenetrating edges, and determine whether an **in** classification is a result of a deteriorating temporary contact or a collision. Given an entity x of s, we define the *incident entities* of x as the set of entities that bound x—for a face, they are the bordering edges and vertices, for an edge they are the two vertices, and for a vertex, there are none. We also define the *adjacent entities* of x as the set of entities that are adjacent to but not incident to x. Thus, x may have adjacent edges and faces, but not vertices. Let $x/y \in \mathcal{R}_{ij}$ and assume w.l.o.g. that $\dim(x) \leq \dim(y)$ and that for simplicity of explanation, x is contained completely in y. Then x and all the incident entities of x may lie inside the other solid just below y, as well as the beginning portions of the adjacent entities of x. Since however, we need to check if a temporary contact region is colliding, the interpenetrations are allowed as long as the projection of x onto the plane (or line) of y, $\Pi_y(x)$, is in y (see Figure 10). If any point of $\Pi_y(x)$ is inside, then x is colliding. If it is outside, it or some of its contact points broke. In either case, the

time that $\Pi_y(x)$ first came into contact with the boundary of y (i.e, the edges or vertices of the face, or the vertices of the edge) must be found. This formulation is straight forward since only a single contact region is involved. However, when two contact regions become adjacent at a line, or more than two adjacent regions become adjacent at a point, the simple projection of x onto y is insufficient to handle the interpenetrations. Consider the two examples in Figure 11 for which the two contact regions are $\{x_1/y_1, x_2/y_2\}$. The indicated interpenetrations are valid, however, if each region is treated independently, then collisions would be inferred. It follows from the example, that the sector between y_1 and y_2 must be included as an allowed region of penetration.

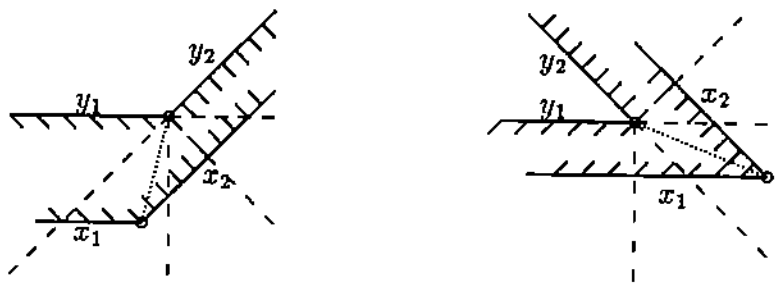

Figure 11: Two examples of allowed interpenetrations when several adjacent regions touch.

Given that either no edges interpenetrate or the ones that do are part of some temporary contact region, we determine the regions of the new $\mathcal{R}_{ij}$, and for each region we compute the contact point pairs. Since there is no friction, it suffices to collect only the points on the convex hull of the region. Since 2D regions need not be convex, the computed points may not all lie on the convex hull and so we throw out the points that are on the inside of the convex hull. This requires $O(n \log n)$ time for n points as described in Preparata's book [23]. First, lets consider how the regions are obtained. The 0D regions are relatively simple to find. They result from either a vertex/x, for any x, or a transversal edge/edge contact. For the first case, each adjacent edge e of the vertex v must have the following classification sequence

$$\mathrm{cl}(e) = [(x,p), \alpha],$$

where (x,p) indicates that v touches x and α is the remaining irrelevant classification sequence of e. The pair of points for this v/x contact are p/q, where p is the point coincident with v as returned by $\mathrm{cl}(e)$, and q is the point computed by projecting p onto the entity x. That is, if x is a face with the support plane satisfying the equation $q \cdot n - d = 0$, we have

$$q = p - (p \cdot n)n - dn.$$

For x an edge with the support line satisfying the equation $q = q_1 + tv$, we have

$$q = q_1 + (v \cdot (p - q_1))v.$$

If the 0D contact region is the result of a transversal edge/edge contact, then the two edges e_1/e_2 must have the classification sequences

$$\begin{aligned} \mathrm{cl}(e_1) &= [\alpha_1, (e_2, p_1), \beta_1], \\ \mathrm{cl}(e_2) &= [\alpha_2, (e_1, p_2), \beta_2]. \end{aligned}$$

and the pair of points returned by the edge classifications is p_1/p_2. Note that in the case of a temporary contact region, either of these could be dug in and the points of contact could be regions of the form

$$\mathrm{cl}(e) = [(\mathbf{in}, p_1), \mathbf{in}, (y, p_2), \alpha],$$

for the adjacent edges of the vertex case, and

$$\begin{aligned} \mathrm{cl}(e_1) &= [\alpha_1, (f_{21}, p_{11}), \mathbf{in}, (f_{22}, p_{12}), \beta_1] \\ \mathrm{cl}(e_2) &= [\alpha_2, (f_{11}, p_{21}), \mathbf{in}, (f_{12}, p_{22}), \beta_2]. \end{aligned}$$

for the edge/edge case. In the later case, each edge penetrates below the other edge piercing its two adjacent faces.

The 1D regions are equally easy to find. They result from either the edge/face or the collinear edge/edge contacts

$$\mathrm{cl}(e) = [\alpha, (x, p), y, (z, q), \beta],$$

where x and z can be either the same as y or any entity incident to y, and α and β may be possibly empty irrelevant sequences. This 1D region has two contact points, one at p on x and the other at q on z.

The 2D regions are harder to compute than the 0D and the 1D regions. Conceptually, each 2D region is enclosed by a sequence of alternating 1D and 0D regions. Therefore, we trace around each region and connect the bordering subregions.

Finally, for each new region x/y in $\mathcal{R}_{ij}$ we compute the contact normal $n(x, y)$. For example, Figure 12 shows the three regions, the contact points and the contact normals for the objects in Figure 3. Regions that persist from previous time step retain their previous normals. By convention, a contact normal points away from s_j. For a 0D region, the normal vectors are computed as follows,

$$\begin{aligned} n(v_i, v_j) &= (\overline{v}_i \times \overline{v}_j) \times \frac{\overline{v}_i + \overline{v}_j}{||\overline{v}_i + \overline{v}_j||} \\ n(v_i, e_j) &= \frac{\overline{e}'_j + \overline{v}_i}{||\overline{e}'_j + \overline{v}_i||} \times \overline{e}_j \end{aligned}$$

$$
\begin{aligned}
n(e_i, v_j) &= -n(v_j, e_i) \\
n(e_i, e_j) &= \pm \overline{e}_i \times \overline{e}_j, \text{ for transversal edges} \\
n(v_i, f_j) &= \mathrm{n}(f_j) \\
n(f_i, v_j) &= -\mathrm{n}(f_j)
\end{aligned}
$$

where $\overline{v}$ is the unit normal that averages out all the face normals adjacent to vertex v, $\overline{e}$ is an edge direction, $\overline{e}'$ is the average of the faces adjacent to edge e, and $\mathrm{n}(f)$ is the normal of the support plane of face f. A normal is in the direction away from the object. Consider now that one of the two objects, say s_j, is a nonmanifold, and the 0D contact region is on a nonmanifold entity x, either a vertex, or an edge. The average of the adjacent faces, (e.g., $\overline{v}$ or $\overline{e}$) may result in a null vector. This results when the manifold components of the nonmanifold at that point cancel out each other. We have to treat the different manifold components separately. Given that x of s_i is a nonmanifold entity with k_i manifold components, and y of s_j is a nonmanifold entity with k_j manifold components, we obtain $k_i k_j$ number of contact normals which we then average.

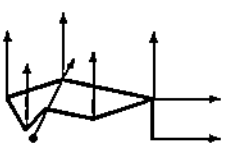

Figure 12: Contact points and their normals (arrows) are shown for the example of Figure 3.

The normal vectors for 1D regions are

$$
\begin{aligned}
n(e_i, e_j) &= \frac{\overline{g}_j - \overline{g}_i}{\left\| \overline{g}_j - \overline{g}_i \right\|}, \text{ for collinear edges} \\
n(e_i, f_j) &= \mathrm{n}(f_j) \\
n(f_i, e_j) &= -\mathrm{n}(f_i)
\end{aligned}
$$

where g_k is the average of the two adjacent face normals of e_k. The 1D region of Figure 12 is an example of the collinear edge/edge contact. For a 2D region the normal is simply

$$n(f_i, f_j) = \mathrm{n}(f_j).$$

11 Discussion

In the early part of 1989, we interfaced the Newton dynamics system to Proto-Solid. Since both systems were written in Common Lisp and ran on the Symbolics

lisp machine, their interface was a simple exercise in establishing communication between two remote modules. At first, no collisions were allowed. The geometric module was used to simply create complex objects and to compute the mass properties. In 1990, we enhanced ProtoSolid with the multidimensional partitioning tree structures to facilitate the detection and analysis of collisions. The structure has proven to be sufficiently efficient to support frequent requests for the static analysis of collisions. Although collisions were straight forward, temporary contact was not. Numerical errors in the integration of the equations of motion could not keep two constrained objects for slightly interpenetrating. This kept creating disagreement between the two modules which caused the simulation to break. Our early attempts at conditionally altering the contact tolerances to keep the interpenetrating in check did not work. Furthermore, we could not recognize special events such as colliding temporary contact regions and breaking geometric contact during interpenetrations. It was then that we decided that contact had to be analyzed in terms of the contact regions and not in terms of individual contact points as we were doing before.

At this point we realize that the way we are doing the dynamics of contact is probably not the best way. Allowing drift and applying correctional forces at contact points may not always produce the results we want. There are methods to satisfy constraint equations to any given accuracy [15]. But those methods have not been applied to systems which account for the the geometry and create new constraints during simulation. We are currently investigating the incorporation of such a method into our system.

Acknowlegements

We have corresponded frequently with Jim Cremer in Cornell on the contact problem. Jim's knowledge and insight into the problem helped us greatly in the writting of this paper. This work was supported by NSF Grant CCR 86–19817.

References

[1] B. Arnaldi and G. Dumont and G. Hegron. "Animation of Physical Systems from Geometric, Kinematic and Dynamic Models." *Proceedings of the IFIP TC 5/WG 5.10 Working Conference on Modeling in Computer Graphics*, Tokyo Japan, April 1991.

[2] D. Baraff. "Curved Surfaces and Coherence for Non-Penetrating Rigid Body Simulations," *Proceedings of ACM SIGGRAPH '90*, 24(4):19–28, August 1990.

[3] J. W. Boyse. "Interference Detection Among Solids and Surfaces," *Communications of the ACM* 22:3–9, January 1979.

[4] S. Cameron and R. K. Culley. "Determining the minimum translational distance between two convex polyhedra," *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 591–596, 1986.

[5] S. Cameron. "Collision Detection by Four-Dimensional Intersection Testing." *IEEE Transactions on Robotics and Automation*, 6(3):291–302, June 1990.

[6] J. Canny. "Collision Detection for Moving Polyhedra," *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 8(2):200-209, 1986.

[7] B. Chazelle and D. P.Dobkin. "Detection is easier than computation." *Proceeedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 146–153, April 1980.

[8] J. F. Cremer, "An architecture for General Purpose Physical System Simulation–Integrating Geometry, Dynamics, and Control." PhD Thesis, TR 89–987, Department of Computer Science, Cornell University, April 1989.

[9] J. F. Cremer and A. J. Stewart. "The Architecture of *Newton*, a General-Purpose Dynamics Simulator", *IEEE International Conference on Robotics and Automation*, pages 1806–1811, 1989.

[10] R. Featherstone. *Robot Dynamics Algorithms*, Kluwer Academic Publishers, Boston, 1987.

[11] M. Gardner. "Mathematical Games," *Scientific American*, 207:120–126, 1962.

[12] E. .G. Gilbert and D. W. Johnson and S. S. Keerthi. "A fast procedure for computing the distance between complex objects in three space." *IEEE Journal of Robotics and Automation*, 4:193–203, April 1988.

[13] E. .G. Gilbert and C-P. Foo. "Computing the Distance between General Convex Objects in Three-Dimensional Space", *IEEE Transactions on Robotics and Automation*, 6(1):53–61, February 1990.

[14] J. K. Hahn. "Realistic Animation of Rigid Bodies," *Proceedings of ACM SIGGRAPH '88*, 22(4):299–308, August 1988.

[15] E. J. Haug. *Computer-Aided Kinematics and Dynamics of Mechanical Systems, Volume I: Basic Methods*, Allyn and Bacon, Boston MA, 1989.

[16] T. Lozano-Perez and M. A. Wesley. "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles." *Communications of ACM*, 22(10):560–570, October 1979.

[17] C. M. Hoffmann and J. E. Hopcroft. "Simulation of physical systems from geometric models", *IEEE Journal of Robotics and Automation*, RA–3(3):194–206, June 1987.

[18] C. M. Hoffmann and J. E. Hopcroft. "Model generation and modification for dynamic systems from geometric data," *CAD Based Programming for Sensory Robots*, F50:481–492, 1988.

[19] C. M. Hoffmann and G.Vaněček, Jr. *Advances in Control and Dynamics*, Chapter: Fundamental Techniques for Geometric and Solid Modeling, C.T.Leondes, editor, Academic Press, 1991.

[20] P. Lötstedt, "Numerical Simulation of Time-dependent Contact and Friction Problems in Rigid Body Mechanics," *SIAM Journal of Scientific and Statisctical Computing*, 5(2):370–393, 1984.

[21] M. Moore and J. Wilhelms. "Collision Detection and Response for Computer Animation," *Proceedings of ACM SIGGRAPH '88*, 22(4):289–298, August 1988.

[22] A. Paoluzzi. "Motion Planning+Solid Modeling=Motion Modeling", *Dip. di Informatica e Sistemistica*, Università "La Saoienza", Technical Report 17–89, Rome, November 1989.

[23] F. P. Preparata and M. I. Shamos. *Computational Geometry, an Introduction*, Springer-Verlag, New York, 1985.

[24] R. .B. Tilove. "A Null-Object Detection Algorithm for Constructive Solid Geometry." *Communications of the ACM*, 27(7):684–694, July 1984.

[25] T. Uchiki and T. Ohashi and M. Tokoro. "Collision Detection in Motion Simulation." *Computers & Graphics*, 7(3–4):285–293, 1983.

[26] G. Vaněček, Jr., "Brep-Index: A Multi-dimensional Space Partitioning Tree," *First ACM/SIGGRAPH Symposium on Solid Modeling Foundations and CAD/CAD Applications*, Austin Texas, 35–44, June 1991.

[27] G. Vaněček, Jr., *ProtoSolid: An inside look*, Purdue University, Department of Computer Science, CER–89–26, November 1989.

[28] G. Vaněček, Jr., "A Data Structure for Analyzing Collisions of Moving Objects," *IEEE Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, Kailua-Kona Hawaii, Vol I:671–680, January 1991.

[29] J. Wittenburg, *The Dynamics of Systems of Rigid Bodies*, B. G. Teubner, Stuttgart, 1977.