1992

Modeling Contacts in a Physically Based Simulation

William J. Bouma

George Vaněček

Report Number:
92-077

MODELING CONTACTS IN A PHYSICALLY BASED SIMULATION

William J. Bouma
George Vanecek, Jr.

CSD-TR-92-077
September 1992
(Revised November 1993)

Modeling Contacts in a Physically Based Simulation

William J. Bouma and George Vaněček Jr.

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract

In this paper[1] we present a model for polyhedral contact that may be used in a physically based simulation. The model is based on a geometric analysis formulated as a static problem in three-dimensional space. The geometric contact analysis between a pair of objects determines the set of surface areas which are in contact and their associated normals. From these areas and normals the dynamics simulator can formulate equations that model the physical consequences of the contact. For areas that come into contact with high relative velocity, these equations model the collision impulse force. The pushing and sliding behavior of low relative velocity contact is modeled by the addition of kinematic constraints over the areas in contact.

The geometric analysis consists mainly of collecting adjacent contact points into separate regions which share the same normals. Since non-convex polyhedra are allowed, the intersection can produce surface areas which have more than one associated normal. The geometric modeler can provide this entire set of normals or any small subset which spans the same space, whichever the dynamics modeler requires. When a contact region persists from time step to time step in the simulation, the contact analysis can return the normals for that region from the previous time step rather than recalculating them. This temporal coherence of the normals can resolve certain cases where the normals of a region are indeterminate.

The interpretation of the contact requires a full set-theoretic intersection of the two objects. The objects in question are assumed to be already in contact or in close proximity. An efficient and robust implementation of the intersection is achieved by using the brep-index data structure in conjunction with a back-face culling technique based on relative velocities at surface points.

1 Introduction

Realistic simulation of the physics of objects in motion requires the prevention of interpenetration between the solid bodies. Through the use of constraint equations and reaction forces, the dynamics simulator can keep surfaces apart. To correctly place these

[1] to Appeared in Computer Aided Design.

constraints as the objects move and interact requires a geometric analysis of the solids. Each pair of objects is intersected to determine the surface regions in contact and the set of normals belonging to the points in these regions. Given these regions and normals, a system of equations is formulated which models the dynamical consequences of the contact. The form that each equation takes is dependent upon the relative velocity of the objects at the contacting areas. Each contact is either determined to be a *collision contact*, which exists for an infinitesimal amount of time, or a *temporary contact*, which persists for a measurable duration. In the case of a collision, the generated equations describe an instantaneous velocity change between the impacting objects. In the case of a temporary contact, new equations are added to the existing set which constrain the relative motion of the objects over the contacting areas.

We consider the use of model-driven rigid-body dynamics simulators, such as *Newton* [6], that have an event handling mechanism for dealing with discontinuities in the simulation, and which perform a geometric analysis of the contact. For such simulators, the contact is analyzed as a static problem at each time step while taking the state of the previous time step into account. From a geometrical point of view, contact analysis takes place when two objects touch (without interpenetrating); it is based on a set-theoretic classification of contact obtained from a set-theoretic intersection of two objects.

Our formulation of contact analysis makes several assumptions:

1. The objects are polyhedral given as boundary representations (B-reps).

   The use of rigid bodies with planar faces allows preprocessing of the objects to improve the efficiency of the contact analysis by the geometric module. Curiously, the restriction that surfaces be planar increases the difficulty of the contact analysis when compared to the analysis of objects with curved surfaces containing no reflex edges or vertices.

2. The objects are rigid bodies.

   For objects that are modeled as polyhedra rather than point masses, contact analysis remains an open problem when deformations occurring during collisions and contact are taken into account.

3. The relative motion of an object during the interval Δt is sufficiently small so that no changes in contact regions are missed. This assumption also holds for detecting collision of two impacting objects.

   Without this assumption, the space-time sweep of all objects would need to be modeled in four-dimensional space to guarantee that all collisions and contact changes are detected.

4. In the initial configuration no two objects may interpenetrate.

   During a simulation, an interpenetration of two objects reveals an undetected collision event. Since no collisions can theoretically take place before the start of the simulation, this condition cannot be resolved.

5. In the initial configuration, no ambiguous geometric contact can occur, as described in Section 3.

Certain contacts can be analyzed only when the state of the previous time step is taken into account. Since there is no state for a previous time step at the start of a simulation, the occurrence of ambiguous contacts in the initial configuration will result in unexpected motions.

Note that situations 4 and 5 can easily be detected during the normal course of the contact analysis. Any software based on our method should report such problems in the initial setup.

Consider for the time being the further assumption of using only objects that are convex. For a pair of touching convex objects, the contact analysis is fairly straight forward. Their contact can be either a single point, a line segment, or a convex polygon. The only difficulty arises in determining the contact normal for the indeterminate cases, such as a vertex-on-vertex, or a vertex-on-edge contact. This problem has been addressed by many, such as [21, 2, 14]. Most, however, treated this as a collision problem and ignored the case of persistent contact. In doing so, the normals are usually incorrectly computed at times when the dimension of a region changes, such as when a 2D region shrinks to a 1D region. To prevent a discontinuity, the authors first suggested in [21] to look at the corresponding normals from the previous time step.

The use of complex (i.e., nonconvex) geometrical shapes greatly complicates the contact analysis. In this paper, we describe a general model for contact to support dynamics simulation systems such as *Newton* [10, 11, 5]. Here we extend the work on the system proposed in [21] and [3]. We are concerned here with analyzing the geometry in a meaningful way for the dynamics. The geometric analysis of contact can only be understood in context of the entire model-driven simulation paradigm, and so in Section 2, we outline the dynamics engine of the *Newton* system. This includes the handling of exceptional events and the issues related to temporary contact analysis. In Section 3 we propose the model for contact based on the analysis of the set-theoretic intersection. In Section 4, the implementation details using the Brep-index and back-face culling are given. The huge number of contact queries and analyses suggests using specialized solid representations that facilitate and optimize such geometric computations. In our implementation of contact analysis, we perform a preprocessing step that attaches a multidimensional space partitioning (MSP) tree structure to each B-rep to serve as a volumetric index into the B-rep. This combined structure is called the *Brep-index* [20]. Its use allows a subquadratic cost for the set-theoretic intersection of objects. Since a null intersection reveals no contact, we can also use it for the initial collision detection. Here, we apply a back-face culling technique [22] to eliminate approximately half of the boundary from having to be checked against the brep-index.

In general, any collision detection approach can be used to first determine that contact is taking place [4, 9]. For example, recent work by Lin and Canny [12] provides a collision detection algorithm for convex objects that is almost constant time per time step, independent of the model complexity. This also generalize to n-bodies. We can approximate all object with their convex hulls and apply such algorithms. However, once the convex hulls touch or interpenetrate, we apply our method to obtain a detailed geometric analysis of the contact.

Note that interesting simulations have objects that are in constant contact. For

instance, articulated bodies, such as robots, walking around and interacting with other objects. In these simulations, the n-body collision detection methods that inform us when two objects are in close proximity do not apply.

2 Preliminaries

Contact analysis takes place in the context of some dynamics simulation model. Although this paper is not about dynamics, it is necessary to know something about the assumed model of dynamics to understand the geometric contact problem. In this section, we briefly review the dynamics engine of the *Newton* simulation system and point out some of the general issues of dynamics simulation which relate to the contact problem.

2.1 Dynamics Engine of a Model-Driven Simulator

In a high-level definition language, an initial configuration is given which defines the objects: their positions and velocities, various constraints and forces upon them, their shape, and other physical properties. The specification of the initial configuration is parsed and yields a set of ordinary differential equations (ODEs). These are based on the *Newton*-Euler equations. The basic motion equations for a single object are:

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

where m is the mass, $\ddot{\boldsymbol{r}}$ is the acceleration of the center of mass, $\boldsymbol{J}$ is the 3-by-3 inertia matrix, $\boldsymbol{F}$ and $\boldsymbol{T}$ are the force and torque on the object, and $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ are the angular velocity and acceleration. Optionally, the motion of two objects may be constrained in relation to each other by the introduction of one or more additional equations. Such a constraint is referred to in *Newton* as a *hinge*. Each positional constraint equation is 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

$$\boldsymbol{r}_i + \boldsymbol{c}_i = \boldsymbol{r}_j + \boldsymbol{c}_j,$$

where $\boldsymbol{r}_i$ is the position of the ith object's center of mass, and $\boldsymbol{c}_i$ is the vector from the center of mass to the location of the hinge point. The second derivative yields the acceleration constraint

$$\ddot{\boldsymbol{r}}_i + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{c}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{c}_i) = \ddot{\boldsymbol{r}}_j + \dot{\boldsymbol{\omega}}_j \times \boldsymbol{c}_j + \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times \boldsymbol{c}_j).$$

The addition of the hinge equation results in an unknown force between the two objects at the contact point. This force, $\boldsymbol{X}_{ij}$, is thus added to the motion equations for object i and subtracted from those of object j. The new equations for object i become

$$\begin{aligned} m_i\ddot{\boldsymbol{r}}_i &= \boldsymbol{F}_i + \boldsymbol{X}_{ij} \\ \boldsymbol{J}_i\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \boldsymbol{J}_i\boldsymbol{\omega}_i &= \boldsymbol{T}_i + \boldsymbol{c}_i \times \boldsymbol{X}_{ij}. \end{aligned}$$

Newton has a wide variety of hinges to restrict the various degrees of freedom between objects and their 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 the spatial volume of an object in motion may come into contact with some other object's volume. At a contact point ρ, a non-negative 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 velocity of a point $\rho = r + c$ on an object is given by

$$\dot{\rho} = \dot{r} + \omega \times c. \tag{1}$$

The relative velocity of the contact points in the contact normal direction, n, 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

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

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.

Non-impact 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. The hinge adds kinematic constraints to keep the objects from interpenetrating. This constraint allows the vertex to slide freely within the face, but does not allow it to dip below the face into 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. We call such hinges that are created and broken automatically by *Newton* during a simulation *temporary hinges*. Occurrences of new contact or changes in temporary contact are known in *Newton* as *exceptional events* because they cause discontinuities in the simulation. Accurate detection and handling of contact change events is essential for realistic simulation.

Consider the simulation loop which iterates over time. At time t, a complete state of the system is obtained, consisting of the positions, velocities, and accelerations of all the objects. The set of ODEs is solved for the accelerations and forces on the objects at t. The velocities and positions are obtained by integration from the accelerations. The loop repeats with t incremented by a prespecified time interval Δt. However, if any exceptional events have been detected during the $[t, t + \Delta t]$ interval, time cannot be advanced the full Δt amount. Instead the simulator isolates the time of the first-event occurrence to within a specified tolerance and advances t to the time of that event. After computing the system state at the new time, it affects any changes to the state caused by the event, and the simulation proceeds.

For example, suppose that after computing the positions of the objects at $t + \Delta t$ two objects are found to be interpenetrating, (i.e. their intersection has non-zero volume). Then there must be a time between t and $t + \Delta t$ at which the surfaces of the objects were in perfect contact. A bisection loop or other root finding method gets close to that time. After updating the simulation state, it is found that the colliding points have negative relative collision velocities. A system of collision equations is formulated and solved. The velocities of the objects are then modified directly to reflect the instantaneous change in velocity caused by the impact.

2.2 Dynamics Simulation Involving Temporary Contact

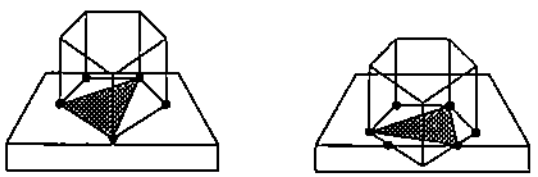

Figure 1: *An active contact point breaks at time t_1 and a new support region (shown shaded) is created at t_2.*

The detection of geometric exceptional events is conceptually simple, but computationally expensive. There are two possible geometric contact events. Either a new contact occurs, or some active contact, (one for which the simulator is maintaining a constraint equation), goes away (see Figure 1). Both of these occurrences are found by doing a set-theoretic intersection of the objects and comparing the found contact regions with the contact regions from the previous time step. The intersection of arbitrary polyhedra is CPU intensive and may need to be performed several times during a single time step of the simulation. A desire to cut the cost of intersection operations is what motivates the final sections of this paper.

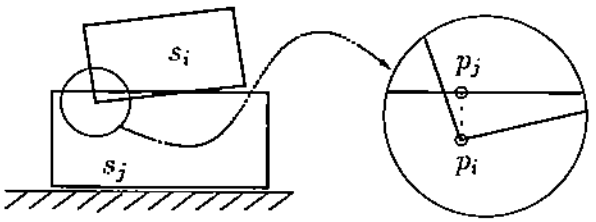

Figure 2: *Positional drift during the temporary contact of two objects.*

Analysis of contact geometry consists of determining the contacting regions and the surface normals over all points in those regions. These normals are the basis of the dynamics constraint equations. A normal at some contact point defines a halfspace relative to one object which the point on the other object is not allowed to enter. Since we are dealing with polyhedral objects the complete contact consists of 0D, 1D, or 2D regions on a plane, 0D or 1D regions on a line, or 0D regions on points, and these need not all be coplanar. An nD region contact is modeled by $n+1$ point-on-plane constraints except in the case where a 0D contact is a result of two transversal edges where it is modeled instead as an edge-on-edge constraint. The difference between these two constraints is that the normal is given by the contact plane in the point-on-plane case, and it is given by the cross product of the two edges in the edge-on-edge case.

To the dynamics simulator, modeling temporary contact is a matter of restricting the degrees of freedom of the contacting objects. A single 0D contact on an object restricts one of its degrees of freedom. When one or both of the contacting objects is non-convex, up to six constraint equations between them can exist. A 2D region of contact is modeled by three point-on-plane constraints. Such a plane-on-plane contact constrains one degree of translational and two degrees of rotational freedom between the objects. Although the 2D region consists of an infinite number of points, only three of these must be chosen on which to place constraint equations. Methods for determining the active kinematic constraints for a geometric contact region are discussed in [1, 2, 7, 13, 5]. However, the problem is not considered to be solved in general. Accurate and robust numerical integration techniques have yet to be developed for handling contact constraints.

Numerical error is a major hindrance in analyzing the contact geometry of objects in a mechanical simulation. In order to analyze the contact, the geometric modeler requires that no point on one object interpenetrate another object to a depth greater than some small constant *epsilon*. Two objects are said to *touch* if they meet this condition, and in addition, some point on one object is within *epsilon* distance of the other. The dynamics simulator must provide the geometric modeler with objects that are either not in contact or touch; no interpenetrations beyond *epsilon* are allowed. But often the systems of constraint equations encountered in mechanical simulation are difficult to solve accurately using common integration methods. Numerical inaccuracies can propagate through subsequent iterations of the simulation loop. Over time the cumulative error causes constrained contact points to drift apart and the objects to interpenetrate arbitrarily (refer to Figure 2).

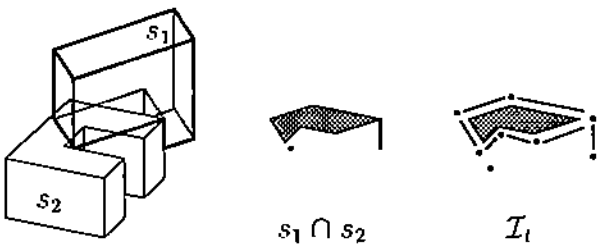

Figure 3: *Two touching objects, their nonregularized intersection, and the set of intersecting entities.*

In particular, *Newton* uses a kind of penalty method to correct for drift between hinged points. A force proportional to the distance and relative-velocity between the hinged points is added in to the constraint equation to push the points back together. With this method, there is no maximum distance beyond which the constrained points are guaranteed not to drift. Any simulator that relies solely on penalty methods to resolve contact, (e.g., [14]), suffers from this same problem.

To ensure that all hinge constraints are met accurately in *Newton*, a stage is introduced in the simulation loop to reduce the error in positions and velocities to a tolerable amount. This stage occurs after the integration, but before the complete contact analysis. A system of equations is formulated based on the kinematic constraints which reflects the desired positions and velocities of the constrained points. This system is then iterated until a desired accuracy is obtained.

3 Modeling Contact

At time t, we are to analyze the contact of two moving and touching objects s_1 and s_2. The two objects need not be convex, and so the contact between them can be quite complex geometrically. This contact can be partitioned into *regions* of points which share certain properties, and which can be obtained directly from a set-theoretic intersection of the two objects. Based solely on these regions, the set of constraints can then be formulated. This section presents a model of contact based on a geometric analysis of the contact.

Each region can be described as a triple (x, y, Π) where x and y are two intersecting entities, one from s_1 and the other from s_2, and Π is the sequence of points defining the shape of the region. An entity here refers to a topological entity of the B-rep (i.e., a vertex, an edge or a face) where the entity is treated as an *open point set*. This way a face, considered as a point set, does not contain the points lying on the boundary (i.e., the edges), and similarly, an edge does not contain its endpoints (i.e., the vertices). We treat x and y both as the label of an entity and as the point set—its meaning clear from context. The sequence of points, Π, describes either a point, a line segment or a polygon, depending on the dimension of the region. Since the dimension of the region cannot always be determined directly from x and y alone, we obtain the dimension from Π. Based on

the number of points, n, the dimension of a region is

$$\dim(r) = \begin{cases} 0 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \\ 2 & \text{if } n > 2, \end{cases}$$

assuming that for $n > 2$ points, no three adjacent points are collinear. For clarity we refer to the set of points covered by a region $r = (x, y, \Pi)$ as $R(r)$.

Lemma 1 *Let s_1 and s_2 be two moving objects which may touch but not interpenetrate, at time t. Their set-theoretic intersection can be described by a set, $\mathcal{I}_t$, of three-tuples (called* regions*), where*

$$\mathcal{I}_t = \{(x, y, \Pi) \mid \begin{array}{l} x \text{ is a topological entity of } s_1, \\ y \text{ is a topological entity of } s_2, \\ x \cap y \neq \emptyset, \text{ and} \\ \Pi \simeq [p_1, \ldots, p_n]\}. \end{array}$$

Proof: Let s_i be given as a B-rep in the form (V_i, E_i, F_i) where V_i is the set of vertices, E_i is the set of edges and F_i is the set of faces. If s_1 and s_2 do not touch $\mathcal{I}_t = \emptyset$. So assume they touch. Since they cannot interpenetrate, every intersecting entity pair $x \in V_1 \cup E_1 \cup F_1$ and $y \in V_2 \cup E_2 \cup F_2$ does so in a plane, on a line, or at a point, and the intersection may consist of one or more connected components. Each connected component then forms a single region, and can by described by its bounding polygon, Π. When a 2D region has one or more holes, the holes are interconnected by introducing bridge edges. ■

We are assuming here that the intersection set $\mathcal{I}_t$ yields homogeneous regions; that is

$$(\forall p, q) \in R(r)\ (\mathrm{N}_{s_1}(p) = \mathrm{N}_{s_1}(q) \wedge \mathrm{N}_{s_2}(p) = \mathrm{N}_{s_2}(q)), \tag{2}$$

where $\mathrm{N}_s(p)$ is the open neighborhood of point p on object s. This property can be guaranteed if both objects are manifolds.

Intuitively, in order to form the dynamic constraints, the simulator must be supplied with the set of all contacting points, and for each, a set of directions it is not permitted to move if penetration is to be avoided. This implied set of kinematic inequality constraints will be referred to as the geometric *constraint* on a point. From Eq. (2) it can be seen that for any region r, each point in $R(r)$ will have the same constraint. Thus we can refer to the constraint on a region.

We show that not all of the regions in $\mathcal{I}_t$ are necessary and derive a subset of $\mathcal{I}_t$ which suffices to geometrically constrain the objects. We denote this subset by $\mathcal{R}_t$. We cannot prove from the geometry alone that all of these regions are necessary to constrain the motion, only a dynamic force analysis can decide that. Thus the dynamics engine may end up throwing away some of the regions of $\mathcal{R}_t$.

To understand $\mathcal{R}_t$, consider the intersection set, $\mathcal{I}_t$, for the two objects of Figure 3. Of the 16 regions in $\mathcal{I}_t$, three contain in their borders the other 13. The three are: a 0D vertex-on-edge, a 1D edge-on-edge, and a 2D face-on-face regions. These regions do not

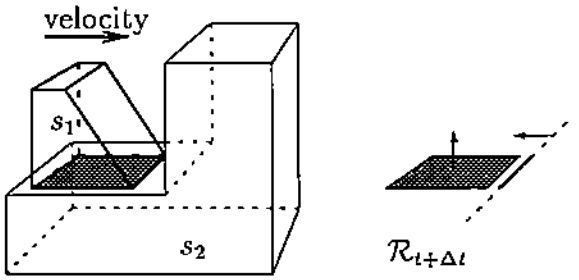

Figure 4: *An example where a temporary-contact region and a collision region are incident; (a) shows time t, and (b) shows the regions at time $t + \Delta t$.*

border any other higher-dimensional regions and they alone are sufficient to describe the constraints between the objects.

In general, to account for more complex constraints arising from the nonconvexity of the objects we may need to include some of the lower-dimensional regions bordering higher-dimensional regions. This can be illustrated by the situation in Figure 4(a) in which object s_1 is sliding across object s_2. The objects are kept apart by a single temporary-contact face-on-face region, and so although $\mathcal{I}_t$ has nine regions, only the 2D face-on-face region is needed to describe the contact. Eventually the sliding of s_1 causes it to collide with a protrusion on s_2. This collision manifests itself as an additional 1D edge-on-edge region adjacent to the 2D region. Therefore, at the time of the collision we must have both the 2D and the adjacent 1D regions; one describing the temporary contact, and the other, the collision (as shown in Figure 4(b)). We say that two entities $x, y \in s$ are *adjacent* if and only if (assuming without loss of generality that $\dim(x) < \dim(y)$)

$$(\exists p) \in x \ (N_s(p) \cap y \neq \emptyset).$$

For example, an edge is adjacent to either one of its two faces, but the two faces of the edge are not. Also, the end vertices of the edge are adjacent to the edge and also to the two faces of the edge.

A region, r, is needed if none of the adjacent higher-dimensional regions fully constrain it. By removing the parts of s_1 and s_2 around r that are already constrained by the adjacent higher-dimensional regions, if anything of s_1 or s_2 is left over in the neighborhood of r then an interpenetration is possible, and so r must be kept. Thus, to determine if a region is needed, the effect of the constraints from the adjacent regions must be taken into consideration. To facilitate this, we introduce the space, $H_i(r')$, to approximate the volume of s_i around a higher-dimensional and adjacent region r' of r. This space is formed by at most two halfspaces since for any r, r' must be either one or two-dimensional. By convention, the halfspaces are oriented so that their normals point away from s_i.

Definition 1 *For a 1D or a 2D region $r = (x, y, \Pi)$, let $H_1(r)$ locally approximate r*

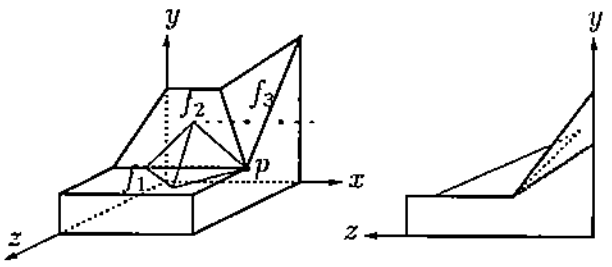

Figure 5: *An isometric and a side view of two objects in contact. Here $\mathcal{R}_t$ contains a 2D region on face f_1, a 1D region on the edge between f_1 and f_2, and a 0D region at point p. The back face of the tetrahedron does not touch f_2.*

around s_1 as

$$H_1(r) = \begin{cases} h_0 & \text{if } \dim(r) = 2 \\ h_0 & \text{if } \dim(r) = 1 \wedge x \text{ is a face} \\ h_1 \cap h_2 & \text{if } \dim(r) = 1 \wedge x \text{ is a convex edge} \\ h_1 \cup h_2 & \text{if } \dim(r) = 1 \wedge x \text{ is a reflex edge,} \end{cases}$$

where h_0 is the halfspace bounded by the support plane of face x, and h_1 and h_2 are halfspaces for which $H_1(r)$ contains the adjacent faces of edge x but not the interior of s_2. The definition for $H_2(r)$ is identical except that we consider y instead of x.

Consequently, we can define a sufficient subset of regions, $\mathcal{R}_t$, of $\mathcal{I}_t$ needed to constrain s_1 from interpenetrating s_2 in terms of the 2D, 1D and 0D regions of $\mathcal{I}_t$.

Definition 2 *Let*

$$\mathcal{R}_t = \mathcal{R}_t^2 \cup \mathcal{R}_t^1 \cup \mathcal{R}_t^0$$

be the set of contact regions *with the d-dimensional regions defined as*

$$\mathcal{R}_t^d = \{r \in \mathcal{I}_t \mid \dim(r) = d \wedge (\forall p) \in R(r)\ (N_{a_1}(p) \neq \emptyset \vee N_{a_2}(p) \neq \emptyset\ )\} \tag{3}$$

where

$$a_i = s_i -^* H_i(r_1) -^* H_i(r_2) -^* \ldots \tag{4}$$

for all higher-dimensional adjacent regions, $r_j \in \mathcal{R}_t^k$, of r (i.e., $k > d$). The binary operator, $-^*$, denotes the regularized difference.

The construction of $\mathcal{R}_t^1$ depends on $\mathcal{R}_t^2$, and similarly $\mathcal{R}_t^0$ depends on both $\mathcal{R}_t^1$ and $\mathcal{R}_t^2$ (refer to Equations (3) and (4)).

Theorem 1 *The set of contacts, $\mathcal{R}_t$, is sufficient to constrain s_1 from penetrating s_2.*

Proof: We show that all remaining regions of $\mathcal{I}_t$ that are not in R_t (i.e., regions in $\mathcal{I}_t - \mathcal{R}_t$) are constrained by regions of $\mathcal{R}_t$. We argue separately for each dimension.

For $d = 2$, Eq. (4) reduces to $a_i = s_i$, causing Eq. (3) to become

$$\mathcal{R}_t^2 = \{r \in \mathcal{I}_t \mid \dim(r) = 2\}.$$

Consequently, all the 2D regions appear in R_t and none can be ignored.

For $d < 2$, let $r \in (\mathcal{I}_t - \mathcal{R}_t)$ such that $\dim(r) = d$. Then since $r \notin R_t^d$, from the negation of the second conjunct of Eq. (3), we get

$$(\exists p) \in R(r)\ (N_{a_1}(p) = \emptyset \wedge N_{a_2}(p) = \emptyset), \tag{5}$$

where a_i is given by Eq. (4). From our assumption that regions are homogeneous (i.e., Eq. (2)), Eq. (5) can be rewritten as

$$(\forall p) \in R(r)\ (N_{a_1}(p) = \emptyset \wedge N_{a_2}(p) = \emptyset).$$

This implies that there is no unconstrained volume of either s_1 or s_2 in the neighborhood of r, so r is unnecessary. ■

Although all 2D regions are used, not all 1D regions are needed. A 1D region is needed if it touches other regions in at most two points, or it lies on a reflex edge (i.e., having a concave sector) of one of the objects and both of its adjacent faces are not already part of some 2D contact region (which is the case for the objects in Figure 4).

Similarly to the selection of the 1D regions, not all 0D regions of $\mathcal{I}_t$ are needed. Consider the example of Figure 5. The need for a 0D region at point p is uncertain. Since the tetrahedron is already constrained by f_1 and f_2, the vertex at point p is free to move in the positive x direction; thus no interference exists between p and f_3. Yet if the tetrahedron moves in the x direction, the neighborhood of p on s_1 will penetrate face f_3. Therefore, a 0D contact region which constrains s_1 from penetrating f_3 is necessary. Detecting the penetration depends on the angles α and β shown in Figure 6. The penetration could not occur if the angle α was larger than the angle β. This example shows that the need for a 0D region does not depend on the 0D region itself but instead on the configuration of its adjacent edges and faces, and on their constraints. We note that if the 1D edge-on-edge region r had $H(r) = h_1 \cup h_4$ as the approximating halfspaces, Eq. (3) would correctly resolve these cases.

Contact Normals: Once the contact regions, $\mathcal{R}_t$, are obtained, the contact normals for each region are computed. How the normals are computed depends on the time that the region first appeared. If the region persists, it is a temporary contact and instead of computing the normals based on the local neighborhoods, the normals persist from the previous time steps. This is to prevent any abrupt discontinuities in the direction of any contact point normal. It is thus necessary to retain the set of temporary-contact regions from the previously consistent time step to correctly compute the normals of the current time step.

Definition 3 *Given $\mathcal{R}_t$, the set of contact regions found at time t, let C_t be the subset of $\mathcal{R}_t$ consisting of the* temporary-contact regions *identified by the dynamics system.*

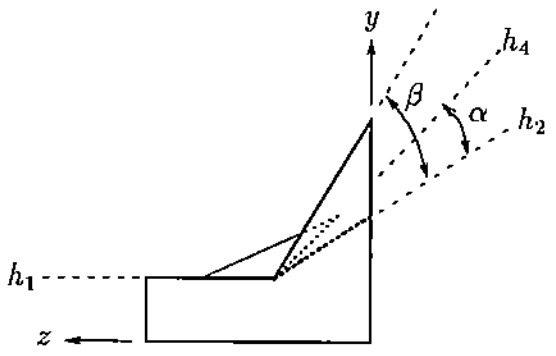

Figure 6: *Computing normals for the 1D region for the example of Figure 5. The collision can exist only when* $\alpha \leq \beta$.

Given $C_{t-\Delta t}$, the temporary-contact regions from the previous time step, the contact normals for the regions in $\mathcal{R}_t$ are computed as follows:

1. Temporary-contact regions that persist from the previous time step retain the same contact normals. A triple $(x, y, \Pi) \in \mathcal{R}_t$ is such a region if

$$(\exists \Pi')\ (x, y, \Pi') \in C_{t-\Delta t}. \tag{6}$$

   Since the region may be moving on the surface, Π need not be the same as Π', and $\dim(\Pi)$ may be different from $\dim(\Pi')$.

2. Temporary contact regions that shrink to lower dimensional regions and that do not collide also retain the same contact normals. Here, a triple $(x, y, \Pi) \in \mathcal{R}_t$ is such a region if the following conditions hold:

$$\begin{gathered}(\exists x', y', \Pi')\ (x', y', \Pi') \in C_{t-\Delta t} \\ (\mathrm{adj}(x, x') \wedge y = y') \vee (x = x' \wedge \mathrm{adj}(y, y')) \vee (\mathrm{adj}(x, x') \wedge \mathrm{adj}(y, y')) \\ \text{both } x' \text{ and } y' \text{ have convex neighborhoods}\end{gathered} \tag{7}$$

   Here $\mathrm{adj}(a, b)$ is a predicate that is true when a is adjacent to b.

3. Regions not found in the previous time step have new contact normals computed based on their local geometries and adjacent contact regions. That is, $(x, y, \Pi) \in \mathcal{R}_t$ is a newly formed region if

$$(\forall \Pi')\ (x, y, \Pi') \notin C_{t-\Delta t}. \tag{8}$$

   For a k-D region, we may have to return at most $3-k$ contact normals. Each contact normal corresponds to a restriction on one degree of freedom from the points in the contact region.

   For a 2D region, the contact normal given is the normal of the support plane of face y.

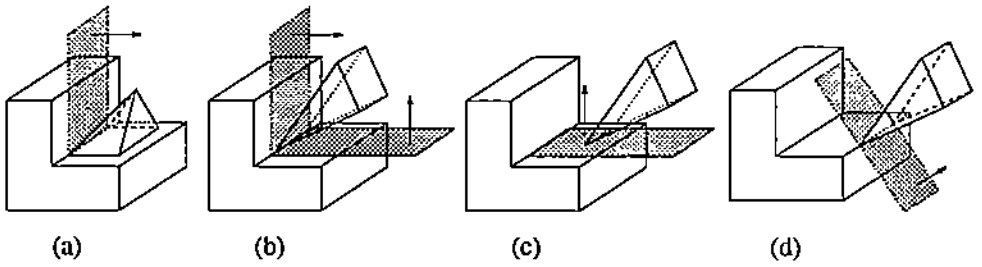

Figure 7: *Examples of contact normals for 1D regions.*

For a 1D region, either one of the edges is reflex or both are convex (see Figure 7). For the reflex-convex case, we return the two normals of the support planes of the two faces adjacent to the reflex edge (as shown in Figure 7(a) and (b)). For the convex-convex case, either we have an edge-face contact (as shown in Figure 7(c)) or an edge-edge contact (as shown in Figure 7(d)). In either case, we return a single normal for a halfspace that separates the two.

For a 0D region, we return at most three normals. The 1D cases easily extend to the similar 0D cases. The one unique situation is a vertex-on-vertex contact where one of the vertices makes a corner. Here there can be many faces meeting at the one point, but at most three normals must be chosen, because at most 3 degrees of translational freedom can be constrained at a point. For best results, we choose the three maximally transversal planes.

The need for the second case is best illustrated by an example. Consider a block sliding at time $t - \Delta t$ as shown in Figure 8(a). The two edge-on-face contacts on the bottom face of the block have an associated contact normal which is the perpendicular vector to the plane of the bottom face. At time t, seen in Figure 8(b), the left bottom edge of the block comes into contact with one of the base supports. The 1D region is a colinear edge-on-edge contact, which according to the local geometric computation can have a different normal from that of the previous edge-on-face contact. In the example the new normal turns out to be in the exact opposite direction to the velocity of the block, as seen in Figure 8(c). Thus there would be a collision at this time 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 $n \cdot v < 0$, where v is the relative velocity vector at the contact point of that normal. Since the edge-on-edge normal is really indeterminate, it is beneficial to choose the normal that satisfies $n \cdot v \geq 0$. 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-on-edge, vertex-on-vertex, or collinear edge-on-edge contacts that will provide normals which give natural behavior in all situations. The normals derived from local geometry provide a reasonable behavior in the case of impact. For temporary contact, we rely instead on coherence between time steps and require that no such special cases occur at the initial time.

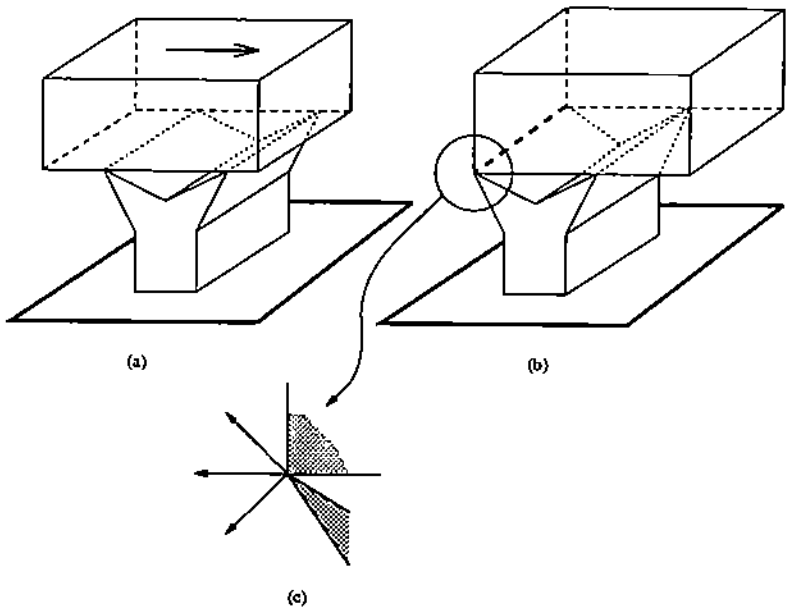

Figure 8: *A block sliding across a two finger support at time $t - \Delta t$, (a), and time t, (b). Figure (c) enlarges the 1D contact region and shows the computed impulse if temporal coherence is not used.*

4 Implementation

At first consideration, our data structures used in the implementation may seem unnecessarly complex. This complexity can be understood from a historical perspective. *Newton* was originally conceived by John Hopcroft and Christoph Hoffman at Cornell circa 1986. The original *Newton* was then written by James Cremer. Up to about 1989, *Newton* dealt with geometry only as parameterized primitives. This included spheres, blocks and cylinders. The advantages were that *Newton* could ignore the exact boundary of primitives and deal with them exclusively on the parametric level—allowing Cremer and Bouma to easily program a collision detection and contact analysis module. The disadvantage of using parameterized primitives was that it allowed only a limited class of solids that could be simulated.

In 1989, Vaněček brought his solid modeling system, *ProtoSolid*, to Purdue from University of Maryland where it was developed. Since both *Newton* and *ProtoSolid* were developed in Common Lisp, Vaněček began to study the problem of integrating *ProtoSolid* to *Newton* to support a broader class of nonconvex polyhedral objects. This was accomplished by packaging *Newton* and *ProtoSolid* into servers and creating a protocol for communicating geometric-event information between the two servers.

The initial difficulty with this system was that *ProtoSolid* was not specifically designed to support a dynamics simulation system, but mechanical design. In the first year, *ProtoSolid* was upgraded to perform mass-property computation and to handle collision detection. The later was done by adding Bruce Naylor's Binary Space Partition (BSP) trees [15]. With the BSP tree support *Newton* could perform simulations with any poly-

hedra, but only for simple contacts. Complex contact models such as the simulation of the tumbling rings failed [3]. This was later found to be due to the insufficient contact information obtainable from the BSP support. Specifically, with the BSP trees, only edge information was possible, and this was insufficient. Consequently, trying to overcome the inefficiency of the BSP trees, Vaněček generalized the trees to multi-dimensional structures and later to the Brep-Index. This led to the model of contact now present in *Newton*. Although the use of the BRep index is not necessary to solve the contact analysis problem, its design was motivated by it.

In this section we describe the algorithm for obtaining the contact regions and for computing the contact normals. The algorithm uses the back-face culling technique to reduce the number of vertices, edges and faces that need to be checked for contact [22], and the Brep-index data structure [20] to obtain the set-theoretic intersection.

4.1 Back-Face Culling

Back-face culling is a preprocessing technique used in computer graphics to speed up the rendering of polyhedra [8]. This technique can be modified to reduce unnecessary checking of boundary elements by the Brep-Index [22]. At any instance of time, some polygons of a moving object are facing in the general direction of motion and some are facing backwards. When considering pairs of objects, the polygons (i.e., faces) of one object that face backwards cannot collide with the other object and these polygons need not be checked for contact. Similarly, edges and vertices whose adjacent faces all face backwards also need not be checked. At each time step, before classifying the entities using the BRep-Index, we determine which entities cannot be part of the contact between the two polyhedra and do not bother with them. Consider how this is done. In the computer graphics technique, the normal vector of a polygon is compared with the view direction. Here, the normal is compared to one or possibly several relative-velocity vectors, and the entity is culled when its motion is in the opposite direction of the normal vector.

The culling technique can speed up the performance of the set-theoretic intersection operation by an average factor of two since roughly half of each object is always facing in the opposite direction of motion. When complex objects consisting of many faces are simulated, this factor of two can provide a noticeable improvement in the performance of the contact-analysis algorithm.

Specifically, consider two objects, i and j, at some time t positioned and oriented in the global frame of reference. Any point $p = \boldsymbol{r}_i + \boldsymbol{c}_i$ on object i has an instantaneous velocity given by Equation (1). The same holds for p and object j. The relative velocity at the point as seen by an observer fixed on object i watching object j is

$$\dot{\boldsymbol{p}}_{ij} = \dot{\boldsymbol{p}}_i - \dot{\boldsymbol{p}}_j, \tag{9}$$

where the instantaneous velocities $\dot{\boldsymbol{p}}_i$ and $\dot{\boldsymbol{p}}_j$ are defined by Eq. (1). Since point p is expressed in terms of both objects i and j as

$$p = \boldsymbol{r}_i + \boldsymbol{c}_i = \boldsymbol{r}_j + \boldsymbol{c}_j. \tag{10}$$

for arbitrary $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, by expanding Equation (9) we get the relative velocity in terms

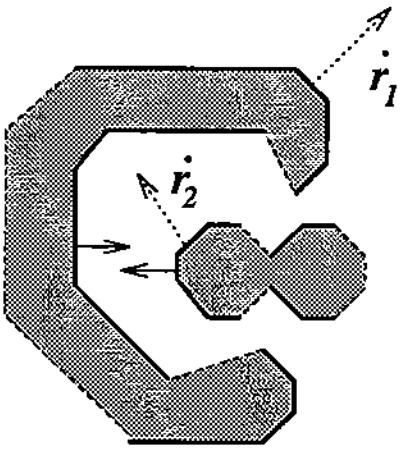

Figure 9: *Simple 2D example showing edges that are culled as dashed lines. For visual simplicity, the objects are not rotating, only translating.*

of the vectors c_i and c_j as

$$\dot{p}_{ij} = \dot{r}_{ij} + \omega_i \times c_i - \omega_j \times c_j, \tag{11}$$

with $\dot{r}_{ij} = \dot{r}_i - \dot{r}_j$. To get the equation in terms of point p, we solve Eq. (10) for both c's, obtaining $c_i = p - r_i$ and $c_j = p - r_j$, and obtain

$$\begin{aligned} \dot{p}_{ij} &= \dot{r}_{ij} + \omega_i \times (p - r_i) - \omega_j \times (p - r_j) \\ &= a_{ij} + p \times \omega_{ji} \end{aligned} \tag{12}$$

where $a_{ij} = \dot{r}_{ij} - \omega_i \times r_i + \omega_j \times r_j$, and $\omega_{ji} = \omega_j - \omega_i$ are constant vectors for a give time t. With this equation, we can compute the relative velocity for any point p, without also having to compute the points c_i and c_j in the relative space of either object i or j. When we think of an object (say, object i) moving in the presence of another object (say object j), we think of object i as having a front and a rear in terms of its relative velocity. We do not expect a collision to occur in the rear of object i. Specifically, let point p be on some Face f of object i in the global frame of reference. The angle θ between the normal vector n_f of face f and $\dot{p}_{ij}$ describes whether p is moving towards the outside directly above f or not. If follows that if θ is less than $\pi/2$,

$$\dot{p}_{ij} \cdot n_f \geq 0,$$

indicating that in the local neighborhood of p, p is moving towards the empty space above f and therefore p can possibly collide with some part of object j within this local neighborhood. Accordingly,

$$\forall p \in f \ (\dot{p}_{ij} \cdot n_f < 0) \tag{13}$$

implies that the entire face is moving away from any portion of object j that may lie directly above it. Therefore f cannot collide and can be culled. It turns out that the relative-velocity vector-space is linear [22], and so it suffices to check only a few points on

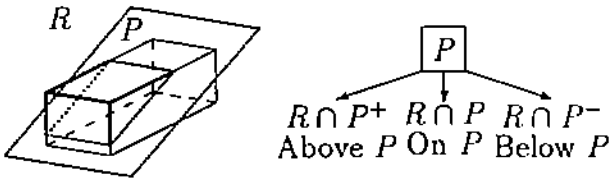

Figure 10: *Region R cut by a plane P is partitioned into three subregions that compose the three subtrees of node representing R.*

the convex hull of the face for this condition. This leads to a constant time test per face. As a simple example, Figure 9 shows two objects in 2D. For simplicity of the illustration, the objects are not spinning. Their velocities are indicated by the dashed arrows. The edges that are moving forward are shown as solid lines, and the edges moving backwards are shown dashed.

Now suppose that we mark each face as being either culled or unculled. We can then cull all edges and vertices based only on these marks. An edge or a vertex is culled if all their adjacent faces are culled.

Note that $\dot{\boldsymbol{p}}_{ij} = -\dot{\boldsymbol{p}}_{ji}$. This means that if p is on face f_i of object i and on f_j of object j, and $\dot{\boldsymbol{p}}_{ij} \cdot \boldsymbol{n}_{f_i} < 0$ it must be the case that $\dot{\boldsymbol{p}}_{ji} \cdot \boldsymbol{n}_{f_j} < 0$, since $\boldsymbol{n}_{f_i} = -\boldsymbol{n}_{f_j}$. Thus both f_i and f_j are culled. This indicates that although contact may exist at time t, it cannot exist instantly after t.

4.2 BRep-Index

The faces, edges and vertices that are not culled by the culling technique have to be tested explicitly for contact. We do this by testing the unculled entities of one of the objects against the brep-index [20] of the other.

The Brep-index consists of a ternary-tree data structure that represents a recursive multidimensional partitioning of space called an MSP tree and a Brep. Associated with an internal tree-node n of the MSP tree is a cut plane P that partitions the region represented by n into three subregions(refer to Figure 10). The three subregions are represented by the three children of n, and are labeled *above*, *on* and *below* P, respectively. By convention, *above* is the halfspace in the direction of the cut plane normal. The resulting spatial partition is called multidimensional because the leaves of the tree represent zero, one, two, and three-dimensional regions. If the region represented by a node is of dimension d, the subregions above and below P are also of dimension d, but the subregion on P is of dimension $d-1$.

If a solid is convex, we can always construct a Brep-index of size $v + e + f$, where v, e, and f are the number of vertices, edges and faces, respectively. For example, a tetrahedron which has four vertices, six edges, and four faces, has a tree with 14 internal nodes. For nonconvex solids, the size grows as a result of the global fragmentation. The MSP trees for a solid are not unique but depend on the ordering and choice of the cut planes. Changing the order results in an equivalent, but different, MSP tree of possibly different size. Two MSP trees are equivalent if the set-theoretic union of the interior

and boundary regions for each tree yields identical solids. A relatively balanced tree can be attained by initially choosing cut planes that split the object roughly in half. As an example, a polyhedral sphere with 128 faces may have an MSP tree with maximum depth of 128 and average depth of 54; such a tree uses only cut planes that are the support planes of the faces. With the addition of initial cuts that are not support planes of the faces, the maximum depth drops to 19 and the average depth to 8.

4.3 The Algorithm

The contact analysis can be separated into several stages. In the first stage, bounding volumes such as spheres are used to quickly determine that two objects are far apart. Two objects, i and j, whose bounding volumes intersect are passed to stage two. Here, the geometrically simpler object has part of its boundary culled and the remaining entities are classified using the Brep index of the other object. This results in the intersection set $\mathcal{I}_{ij}$. In the third stage, the contact regions of $\mathcal{R}_{ij}$ are identified, and the contact normals are computed.

Algorithm: *FindContacts.* Given objects i and j (i.e., s_i and s_j), the transformation matrices M_i and M_j for time $t + \Delta t$, and the previously known contacts C_t, return the newly found contacts $\mathcal{R}_{t+\Delta t}$.

1. Let $M_{ij} = M_i^{-1} \cdot M_j$ be the 4-by-4 transformation matrix that maps object i into the local space of object j at time $t + t_\Delta$, and let $s' = M_{ij} \cdot s_i$.

2. Check the relative velocities of all vertices, edges and faces of solid $M_i \cdot s_i$ and cull the ones known to be moving away from the other object.

3. Classify the remaining entities of s' using the Brep-Index for object j rooted at node n resulting in the set of all contact regions $\mathcal{I}_{t+\Delta t}$:

$$\mathcal{I}_{t+\Delta t} = \bigcup_x \text{Intersect}\big(n, \{(x, \emptyset, \Pi_x)\}\big) \tag{14}$$

   where x is over all the unculled entities of s_i. For each x, the initial bounding points are taken directly from x as Π_x.

```
recursive function Intersect(Node n,
                             Set of Regions R)
  if R = ∅ then
    return ∅.
  else if n is a leaf node then
    If n does not represent an entity,
    then skip the rest.
    Otherwise let y be the entity of n, and
    change each tuple (x, ∅, Π) in R to
    (x, y, Π), and return R.
  else
```

$(\mathcal{R}_a, \mathcal{R}_o, \mathcal{R}_b) \leftarrow$
SplitRegions($\mathcal{R}$, CutPlane(n)).
return
MergeRegions(Intersect($\mathcal{R}_a$, Above(n)),
Intersect($\mathcal{R}_o$, On(n)),
Intersect($\mathcal{R}_b$, Below(n))).

Function SplitRegions takes a set of regions and splits them into subregions lying completely above, on, or below the cutting plane [19].

Function MergeRegions takes the classified regions and performs the following:

(a) Regions classified as lying outside the solid are ignored and not returned.

(b) Adjacent regions with the same classification are merged and returned. This is the inverse operation of SplitRegions.

4. Using $\mathcal{I}_{t+\Delta t}$, create the set of all contact regions $\mathcal{R}_{t+\Delta t}$ by selecting regions described by Equation (3).

5. Using $\mathcal{C}_t$, for each region in $\mathcal{R}_{t+\Delta t}$ compute the contact normal, as described by Conditions (6), (7), and (8).

The Brep-index is specifically designed to supporting efficient and robust classification problems (such as point, line and plane solid classifications), and in this case this helps in performing the set-theoretic intersection. The efficiency comes from having the Brep entities spatially ordered. Consequently, it is not necessary to examine all entities to determine where contact exists. The robustness comes from having the MSP tree for one solid impose a consistent order for classifying the other solid. Step 2 of the algorithm is clearly the most complex. Step 3 takes linear time given the size of the set $\mathcal{I}_{t+\Delta t}$, and Step 4 takes quadratic time given the size of $\mathcal{R}_{t+\Delta t}$ and $\mathcal{C}_t$.

An important part of the geometric analysis is determining where contacts should be for the positional correction phase. That is, numerical error may have caused points to drift, so that a point constrained to lie on a face may no longer lie on that face. The Brep-index allows us to easily classify the various regions to determine how they have drifted.

The Brep-index and Function Intersect have been implemented in C++ as part of a package called Proxima [18].

5 Conclusion

We have made several assumptions that should be reviewed. We assumed that the objects are rigid bodies, and therefore, no deformations occur. It is unclear whether this assumption makes the problem of contact analysis harder or easier. One can argue that if the objects were allowed to deform slightly at the boundary, then all lower dimensional entities (i.e., vertices and edges) could be ignored. This is because only surface on surface

contact could ever exists. This would certainly simplify the contact analysis conceptually and supposedly the determination of surface normals as well. However, it would not reduce the computational complexity of the problem, since nonlinear and locally moving surfaces would have to be modeled. Being that this would be done with patches to reduce the degree of the surfaces, a patch subdivision would be necessary to increase the degrees of freedom around the regions of deformation. The normal computation would also become more difficult since a contact region would have varying normals instead of only a small finite set of normals. Another problem is that during deformations, regardless of how small, self intersections would have to be checked. Thus, removing the rigid body assumption does not simplify the problem. This of course assumes modeling flexible objects as volumes rather than as point masses, where contact can be handled by penalty methods.

We also assumed for the sake of efficiency that the rigid bodies be composed of planar faces. We know how to preprocess planar polyhedra into a multi-dimensional space partitioning structure. We do not know how to do it for nonplanar surfaces; even restricted surfaces like quadrics. The problem of converting a Brep into a Brep index (i.e., by constructing an MSP tree) is analogous to converting a boundary to a Constructive Solid Geometry (CSG) tree. When one introduces nonplanar surfaces then one runs into the separation problem as described by Shapiro and Vossler [16, 17]. To build such an MSP tree, you must find auxiliary planes that correctly partition the differently classified multi-dimensional regions, and we do not know how to do this. Furthermore, should we create such trees, the classification would become much more complex since at each visited node in the MSP tree a trimmed-surface/surface intersection would have to be computed. It is not clear that this would be efficient (the order would remain the same as for the planar case, but the constant would become very large).

We further assumed that the relative motion of an object during any interval is small. One can argue that this is an unnecessary assumption imposed by the discrete-time analysis approach, and that larger time steps can be taken if an other approach was chosen. For example, intersecting four-dimensional objects in time-space or formulating equations that predict the time of the next event when solved. The 4D approach may work for collision detection where objects do not touch for long, but it does not work for prolonged contact. The predictive approach has more merit, but here as well, it cannot replace the discrete time-stepping. We do not know how to completely convert the geometric boundary information into a set of equations. Even if that was possible, the size of the equations would be prohibitive.

There is also the question of friction. Here we assumed no friction, although we also mean quasi-static friction. That is, an object does not slide when there is a compressive force between it and the touching object. This assumption is due to the original dynamics code in *Newton*, and not to inadequacy of the geometric analysis. Without friction, the exact shapes of the contact regions can be ignored. Only the convex hull of each region is necessary. For friction, we need to take the shapes into account. The contact model that we presented here does capture the shapes of the contact regions, and therefore it can support dynamics with friction.

In retrospect, the complete interaction between the geometric and dynamics components of a robust simulator is not perfectly understood at this time. In general, it is not

completely clear where the changes in a temporary contact are recognized; the geometric module or the dynamics module. It is here that we have layed a foundation for future work. A simulator that handles contact should be designed with contact in mind rather than adding contact as an afterthought to collision handling.

We have presented a consistent and reliable method to determine the geometry of contact for mechanical simulation of arbitrarily shaped, nondeformable polyhedra. We have expanded on our previous work by allowing objects to be nonconvex and by looking at the problem from the point of view of contact analysis and not collision detection.

Acknowlegements

This work was supported by NSF Grant CCR 86–19817. We greatly appreciate the numerous discussions on this subject with James Cremer at University of Iowa.

References

[1] D. Baraff, "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies", *ACM Computer Graphics (SIGGRAPH '89)*, 23(3):223–231, July 1989.

[2] D. Baraff, "Curved surface and coherence for non-penetrating rigid body simulation", *ACM Computer Graphics*, 24(4):19–28, 1990.

[3] W. Bouma and G. Vaněček, Jr. "Collision Detection and Analysis in a Physical Based Simulation", *Proceedings of the Eurographics Workshop on Animation and Simulation*, 191–203, September 1991.

[4] J. F. Canny, "Collision detection for moving polyhedra", *IEEE Transactions of PAMI*, 8:200–209, 1986.

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

[6] 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.

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

[8] J. D. Foley, A. van Dam, S. K. Feiner and J. F. Hughes. *Computer Graphics, Principles and Practice*, Second Edition, Addison-Wesley, 1990.

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

[10] 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.

[11] 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.

[12] M. C. Lin and J. Canny. "Efficient algorithms for incremental distance computation", *IEEE Conference on Robotics and Automation*, 1991.

[13] 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.

[14] M. Moore and J. Wilhelms, "Collision Detection and Response for Computer Animation", *Computer Graphics*, 22(4):289–298, 1988.

[15] B. Naylor. "Binary Space Partitioning Trees as an Alternative Representation of Polytopes," *Computer–Aided Design*, 250–252, 1990.

[16] V. Shapiro and D. Vossler. "Brep to CSG Conversion I: Construction and Optimization of CSG Representations", Technical Report CPA89-3a, Mechanical and Aerospace Engineering, Cornell University, 1990.

[17] V. Shapiro and D. Vossler. "Brep to CSG Conversion II: Efficient Representations of Planar Solids", Technical Report CPA89-4a, Mechanical and Aerospace Engineering, Cornell University, 1990.

[18] G. Sun and P. Van Vleet and G. Vaněček, Jr. "Proxima, a Polyhedral Distance and Classification Support in C++", Department of Computer Science, Purdue University, Technical Report, CER–92–41, November 1992.

[19] G. Vaněček Jr., "Set Operations on Polyhedra using Decomposition Methods." Ph.D. Thesis, University of Maryland, College Park, Maryland, June 1989.

[20] 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.

[21] 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.

[22] G. Vaněček, Jr., "Back-Face Culling applied to Collision Detection," to appear in the *Journal of Visualization and Computer Animation*, Daniel Thalmann ed., also available from Department of Computer Science, Purdue University, Technical Report CER–93–11, March 1993.