Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation

David Baraff

Program of Computer Graphics
Cornell University
Ithaca, NY 14853

Abstract

A formulation for the contact forces between curved surfaces in resting (non-colliding) contact is presented. In contrast to previous formulations, constraints on the allowable tangential movement between contacting surfaces are not required. Surfaces are restricted to be twice-differentiable surfaces without boundary. Only finitely many contact points between surfaces are allowed; however, the surfaces need not be convex. The formulation yields the contact forces between curved surfaces and polyhedra as well. Algorithms for performing collision detection during simulation on bodies composed of both polyhedra and strictly convex curved surfaces are also presented. The collision detection algorithms exploit the geometric coherence between successive time steps of the simulation to achieve efficient running times.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

Additional Key Words and Phrases: dynamics, constraints, simulation

1. Introduction

One of the most difficult behaviors to simulate in rigid body dynamics is the non-penetration constraint between solid bodies. The two problems involved in simulating non-penetrating rigid bodies are (1) detecting collisions and contact between pairs of bodies and (2) determining the contact forces present between contacting bodies.

The force determination problem can be solved by analytical and non-analytical methods. Both Barzel and Barr[3] and Baraff[1] give motivations for preferring analytical methods over non-analytical methods in rigid body simulation. Analytical formulations for the contact forces that arise between polyhedral bodies have been presented in [1, 6, 11, 12]. These formulations are the most general possible in that they express the contact forces between bodies that are completely unconstrained in their tangential (sliding) movement. For curved surfaces, formulations for the contact forces that arise to prevent inter-penetration have only been realized for certain cases of constrained tangential movement. For example, if two curved surfaces are restricted to roll without slipping, the contact force between them is easily determined[10, 15]. Similar restrictions such as rolling with a specified slip velocity also have simple analytical solutions. The general case of non-penetration between curved surfaces without any constraint on the tangential movement poses a much more difficult problem; the extension from polyhedra to curved surfaces is not straightforward. We present a formulation of the contact forces between curved surfaces that are completely unconstrained in their tangential movement. We have not encountered a formulation for this problem in any previous literature. We restrict ourselves to the case of twice-differentiable curved surfaces without boundary that contact at only finitely many points. Configurations that result in one- or two-dimensional contact regions are not dealt with. The surfaces need not be convex in the neighborhood of a contact point and may be defined by either implicit or parametric equations. A formulation of the contact forces between curved surfaces and polyhedra is derived as a special case of contact between two curved surfaces.

The collision detection problem has an extensive literary background in the fields of computational geometry and robotics. Computational geometry focuses on problems posed in terms of a static environment. Robotics generally focuses on problems posed in terms of a dynamic environment; the movement of bodies is known in terms of some function of time. In both cases, the emphasis is on developing the best algorithm for solving a *single* problem, either static or dynamic. In contrast, dynamic simulation involves the solution of a *sequence* of static problems, one per time step. Although each problem of the sequence can be solved separately using previous collision detection methods, algorithms specifically designed to solve a sequence of related problems are more efficient. We present efficient collision detection algorithms for polyhedra and convex closed curved surfaces by exploiting the geometric coherence between successive collision detection problems of the simulation. Surfaces can be defined either implicitly or parametrically.

2. Overview

Simulation of non-penetrating rigid bodies by analytical methods involves the basic flow of control shown in figure 1. At every time step, bodies are examined pairwise for possible inter-penetration. If two bodies are found to inter-penetrate, the simulator backtracks to the point in time immediately before the inter-penetration occurred. Once a configuration without inter-penetration is achieved, the contact points between all the bodies are found. Finally, a system of constraint equations based on the contact points yields the analytically correct contact forces and impulses at every contact point. After the contact forces and impulses are applied to the bodies, a new time step is begun. We group this series of steps into two phases: *contact determination* (steps A-D of figure 1) and *force determination* (step E of figure 1) For curved surfaces, both the inter-penetration check and the backtracking steps of the collision determination phase make use of the same mathematical derivations used in the force determination phase. Accordingly, we will deal first with extending

Author address (May 1994): David Baraff, School of Computer Science, Carnegie Mellon University, Pittburgh, PA 15213, USA.
Email: baraff@cs.cmu.edu

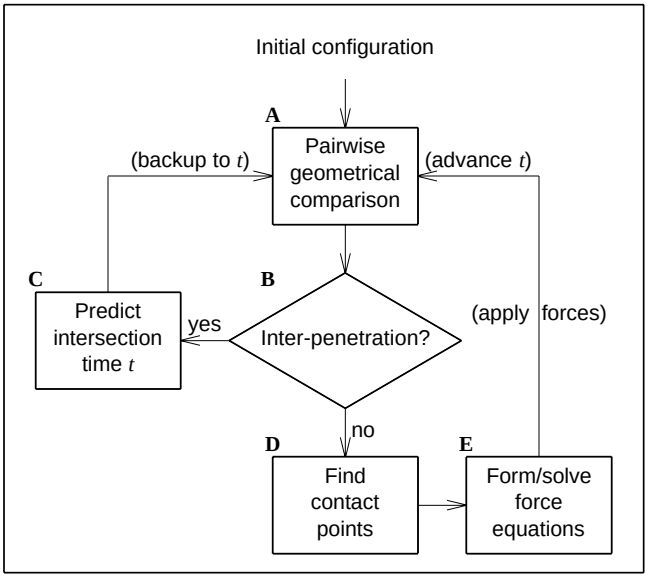

Figure 1. Simulator control flow.

the force determination model to curved surfaces.

3. Analytical Force Determination

Featherstone[6] gives a complete derivation of the system of constraint equations used to find contact forces between rigid bodies. We present a brief review of the mathematical structure of the problem.

At some time t_0 we are given a collection of non-penetrating perfectly rigid bodies that contact at some number of points and asked to calculate the forces between the objects that would naturally arise to prevent inter-penetration. Contact points at which bodies are colliding give rise to contact impulses. Methods for calculating the contact impulses between bodies of any geometry are given in [1, 6, 9]. Contact impulses are calculated and applied prior to considering contact forces. Accordingly it is assumed that the configuration of bodies being analyzed has no colliding contacts.

Consider a contact point p_c between two bodies A and B at time t_0 (figure 2). Let the unit surface normal at the contact point be $\hat{n}$ (see Baraff[1] for the case when $\hat{n}$ is not well defined). In the absence of friction, the (as yet) unknown contact force F is written $F = f\hat{n}$ with f the unknown (scalar) contact force magnitude at time t_0.

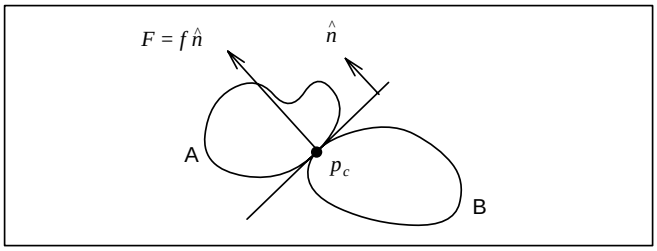

Figure 2. Contact between two objects.

The primary consideration for calculating the unknown contact force magnitude f at each contact point lies in the non-penetration constraint between bodies. At every contact point p_c between two bodies A and B at time t_0, a geometrical constraint that prevents A and B from inter-penetrating near p_c is constructed. Each constraint is converted by a differentiation operation into a constraint on the contact force magnitude f at p_c. The geometrical constraint is expressed by a *constraint function* (or *characteristic function*) $\chi(t)$. A characteristic function $\chi(t)$ is a function of time that characterizes the geometric relation of A and B close to p_c at times near t_0.[1] χ takes on values as follows:

$$\chi(t) = \begin{cases} a > 0 \text{ if } A \text{ and } B \text{ are separate near } p_c \text{ or} \\ a = 0 \text{ if } A \text{ and } B \text{ are touching near } p_c \text{ or} \\ a < 0 \text{ if } A \text{ and } B \text{ are inter-penetrating near } p_c. \end{cases} \quad (1)$$

Given χ, we can express the constraint that A and B not inter-penetrate near p_c as

$$\chi(t) \geq 0. \quad (2)$$

For example, consider a potential vertex-face contact between two planar polygons (figure 3).

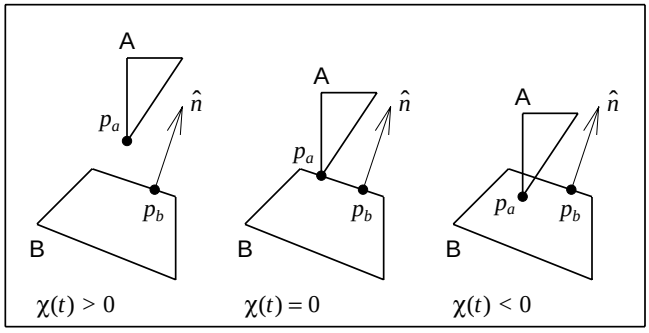

Figure 3. Vertex-face contact.

Let $p_a(t)$ be the position of the vertex of A, $p_b(t)$ be the position of any fixed point on the contact face of B and let $\hat{n}(t)$ be the outwards unit normal of the contact face. The characteristic function for this situation is

$$\chi(t) = \hat{n}(t) \cdot (p_a(t) - p_b(t)). \quad (3)$$

From figure 3, $\chi(t)$ is positive, zero, or negative according to whether $p_a(t)$ lies above, on, or below the contact face of B at time t.

Intuitively, χ may be regarded as a measure of the *distance* between A and B, with negative distance indicating inter-penetration. Similarly, $\dot{\chi}$ may be regarded as a measure of the relative *velocity* between A and B. If $\dot{\chi}(t_0) < 0$ then χ is decreasing and the bodies are colliding; however, it was assumed that the configuration had no colliding contact points. Likewise, if $\dot{\chi}(t_0) > 0$ then χ is increasing and the bodies are separating; in this case the contact force is automatically zero and the contact point may be disregarded. Thus, the only contact points considered are those for which $\dot{\chi}(t_0) = 0$.

In order to convert the geometric constraint $\chi(t) \geq 0$ (equation (2)) into a constraint on the contact forces, we take the second derivative of χ with respect to time and require that

$$\ddot{\chi}(t_0) \geq 0. \quad (4)$$

Informally, we have constrained the relative "acceleration" $\ddot{\chi}$, between A and B, to be non-negative. (Strictly speaking however, $\ddot{\chi}$ is *not* a physical measure of acceleration). Analytically, since $\chi(t_0) = \dot{\chi}(t_0) = 0$, $\ddot{\chi}(t_0) < 0$ would make χ a decreasing function at t_0. This would violate the constraint of equation (2).

[1] We stress the fact that χ is a *local* function and need only be valid for an arbitrarily small open neighborhood of t_0.

How does $\ddot{\chi}(t_0) \geq 0$ constrain the contact forces? While $\chi(t_0)$ and $\dot{\chi}(t_0)$ are independent of any internal or external forces, $\ddot{\chi}(t_0)$ is a linear expression of the contact forces at time t_0. Intuitively, the contact forces must be "strong enough" to satisfy equation (4), and thus prevent A and B from accelerating towards each other at p_c. Appendices A and C show that equation (4) is a linear inequality constraint on the contact force magnitudes.

In addition to the geometrically motivated constraint $\ddot{\chi}(t_0) \geq 0$, there is an additional relationship between $\ddot{\chi}(t_0)$ and f that must be satisfied. If $\ddot{\chi}(t_0) > 0$, then χ is an increasing function at time t_0 and A and B are separating at p_c. In this case the contact force is zero. However, if $\ddot{\chi}(t_0) = 0$ then A and B are not separating and f need not be zero. The relationship between f and $\ddot{\chi}(t_0)$ is known as a *complementarity* condition; it is written as

$$f\,\ddot{\chi}(t_0) = 0 \tag{5}$$

to express the fact that either f or $\ddot{\chi}$ is zero. If we impose the restriction that f be non-negative, so that objects can "push" but not "pull" on each other, then a configuration with N contact points must satisfy the system of equations

$$\ddot{\chi}_i(t_0) \geq 0, \quad f_i\,\ddot{\chi}_i(t_0) = 0, \quad f_i \geq 0 \qquad (1 \leq i \leq N) \tag{6}$$

where f_i and χ_i are the contact force and constraint function for the ith contact point. In a previous paper, we proposed a heuristic method for solving equation (6). For the case of frictionless contact, equation (6) forms what is known as a positive semidefinite (PSD) linear complementarity problem. Equation (6) can also be viewed as a PSD quadratic programming problem. Efficient numerical algorithms exist that solve PSD linear complementarity problems[2] and PSD quadratic programs[14], and we advocate their use over the heuristic solution method. However, in the presence of friction, it is known that equation (6) is no longer necessarily PSD. Finding the solution of a non-PSD linear complementarity problem or quadratic program is NP-hard[14]. A recent result[2] shows that finding a solution to equation (6) in the presence of friction is also NP-hard. Thus, heuristic solution methods may indeed be necessary for practical simulations. For both PSD and non-PSD systems, coherence based methods can be exploited[12, 14] to reduce the computational expense of solving equation (6). See Lötstedt[11, 12] and Featherstone[6] for further discussions on the properties of this constraint system and methods for solving it.

4. Analytical Forces between Curved Surfaces

In the case of polyhedral objects, the characteristic function χ and its second derivative $\ddot{\chi}$ are readily available. Additionally, contact between polyhedra may result in line segments or polygons of contact. Although this results in an infinity of contact points, constraint functions need be formulated only for the finitely many vertices of the convex hull polygon of the contact line or area[1]. For curved surfaces however, the convex hull of the contact area may not be a polygon. For example, a cylinder standing upright on a plane has a circular area of contact points. The convex hull of this contact region is a circle and cannot be described by a finite number of vertices. We have not developed a constraint for contact regions of dimension one or higher. Although discretization of the boundary of the contact area is one possibility, we would rather deal with an analytical formulation over the entire boundary. For curved surfaces, we will restrict our attention to situations in which the number of contact points is finite. We will construct a characteristic function χ for each contact point.

[2] Our simulator uses an implementation of Lemke's algorithm described in [17].

The difficulty in formulating a geometric constraint function for curved surfaces in contact is the need to construct a formula specific enough to be differentiable. How can we formulate the geometric constraint that all points on surface A near a point p_c remain on or outside surface B? Furthermore, how can we write this as a scalar-valued differentiable function that is positive, zero, or negative according to whether A and B are disjoint, contacting, or inter-penetrating? One possible start is to let χ be the minimum distance between A and B near p_c, and require that χ always be non-negative. The minimum distance is positive when A and B are separate and zero when they are contacting. However, as figure 4 shows, the minimum distance is not negative when A and B initially inter-penetrate; it is zero.[3] Additionally, the minimum distance is not differentiable at the time that A and B first contact at p_c.

Closely related to the minimum distance however is the concept of an *extreme* distance. The extreme distance between A and B near p_c is defined as follows. If A and B are disjoint near p_c then the extreme distance between A and B is just the normal minimum distance between A and B (near p_c). If A and B are in contact at p_c, then the extreme distance is zero. If A and B have inter-penetrated near p_c, then the extreme distance is the *maximum* distance between A and B (near p_c). The *extremal points* of A and B are the two points p_a and p_b on A and B that realize the extremal distance (figure 4).

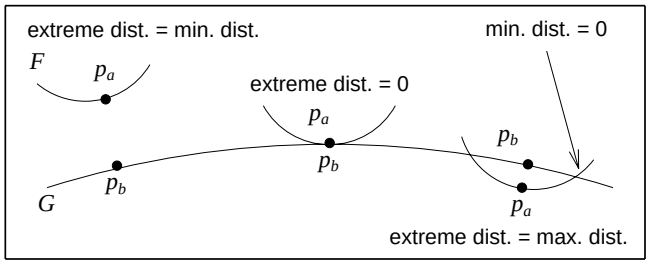

Figure 4. The extremal distance and extremal points.

Given the above definitions, we can construct a constraint function by letting $\chi(t)$ be a positive multiple of the extreme distance when A and B are disjoint or contacting. When A and B have inter-penetrated, we will let $\chi(t)$ be a negative multiple of the extreme distance. In the next section we will show how p_a and p_b can be used to construct such a formula. The restriction to situations where only finitely many contact points arise guarantees that the extreme distance (and the extremal points) of A and B sufficiently near p_c will be unique.

5. Deriving $\ddot{\chi}$

We still must construct an explicit formula for χ so that it may be differentiated to find $\ddot{\chi}$. Although the χ we develop in this section yields an impractical result (computationally speaking), we feel its presentation is necessary to clearly understand the final form of χ and $\ddot{\chi}$ derived in section 6. The derivation of χ and $\ddot{\chi}$ will assume implicit definitions of the curved surfaces; however, the end result depends solely on the derivatives of the surfaces at the contact points. As a result, parametric definitions of the surfaces can be used as easily as implicit definitions for determining contact forces; see appendix D for details.

We model the two curved surfaces of A and B as implicit time-varying functions $F(p,t)$ and $G(p,t)$ where p is a point in world space. At time t, a point p is on the surface of A iff

[3] If one shape lies completely inside the other, the minimum distance between them is positive, but intersection between the surfaces must occur first.

$F(p,t) = 0$. (We will refer to F and G as both functions and surfaces). Furthermore, if $F(p,t) < 0$ then p is inside A, and if $F(p,t) > 0$ then p is outside A. The function G similarly defines the shape of B. We will use the notation

$$\frac{\partial}{\partial p}F(p,t) = F'(p,t) = \nabla F(p,t)^T, \tag{7}$$

where $\nabla F(p,t)$ is a column vector (and $F'(p,t)$ a row vector). If $F(p,t) = 0$, then $\nabla F(p,t)$ is the outwards directed surface normal of F at point p at time t (figure 5).

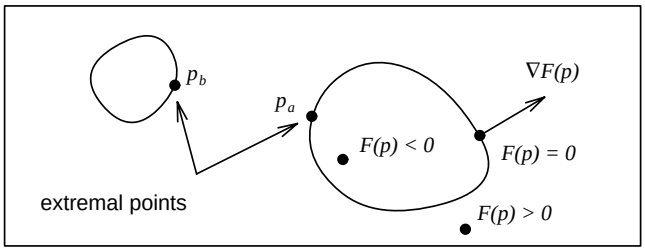

Figure 5. Implicit surface description of A and B in terms of F and G.

Given these definitions, we can express χ as

$$\chi(t) = \nabla G(p_b,t)\cdot(p_a(t) - p_b(t)) \tag{8}$$

where p_a and p_b are the two extremal points between A and B at time t. From figure 6, we see that $\nabla G(p_b,t)$ is colinear with the vector $p_a - p_b$.

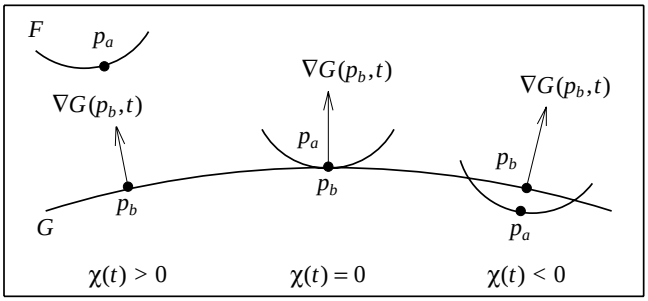

Figure 6. χ expressed in terms of the extremal points.

When A and B are disjoint, $\nabla G(p_b,t)$ is pointed in the same direction as $p_a - p_b$; hence $\chi(t)$ is a positive multiple of the distance $\|p_a - p_b\|$. Similarly, when A and B inter-penetrate, $\nabla G(p_b,t)$ points in the opposite direction of $p_a - p_b$ and $\chi(t)$ is a negative multiple of the distance $\|p_a - p_b\|$. Thus, $|\chi(t)|$ is the extreme distance at time t scaled by $\|\nabla G(p_b,t)\|$ and equation (8) defines a valid characteristic function χ.

The derivative of equation (8) requires the derivatives of p_a and p_b. Although the curved surface characteristic function appears similar to the polyhedral characteristic function (equation (3)), the latter is easily differentiated while the former is not. In the polyhedral characteristic function, the points p_a and p_b denote fixed points of A and B. The derivatives of a fixed point of a rigid body are simply expressed in terms of the motion of the body[3, 9]. However, in the case of curved surfaces, p_a and p_b are not fixed points of A and B. p_a and p_b change positions in two ways. First, p_a and p_b move according to the rigid body motion of A and B. Second, p_a and p_b change positions in the body space of A and B (figure 7). In order to differentiate p_a and p_b, they must be redefined in such a way that they can be differentiated.

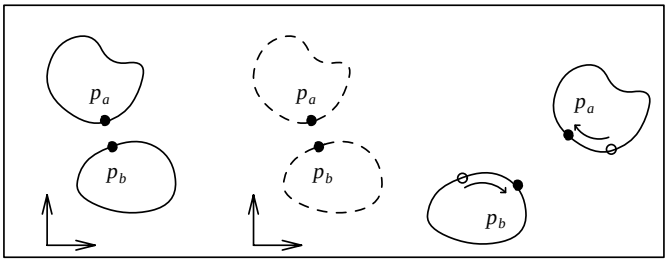

Figure 7. Movement of p_a and p_b in both world and body space.

We define the extremal points p_a and p_b at time t as the (unique) pair of points near p_c that satisfy the conditions

$$\begin{cases} E_1\colon\ \nabla F(p_a,t) + \lambda_2 \nabla G(p_b,t) = \vec{0} \\ E_2\colon\ F(p_a,t) = 0 \\ E_3\colon\ G(p_b,t) = 0 \\ E_4\colon\ (p_b - p_a) + \lambda_1 \nabla G(p_b,t) = \vec{0}. \end{cases} \tag{9}$$

λ_1 and λ_2 are unconstrained scalar values. Condition E_1 guarantees that the surface normals at the extremal points are colinear. Conditions E_2 and E_3 guarantee that p_a and p_b are points on A and B, and condition E_4 guarantees that p_b's displacement from p_a is colinear to the surface normals. A formal justification of conditions E_1 thru E_4 as a definition of the extremal points may be found in any advanced calculus text; see for example Taylor and Mann[16]. Figure 6 shows the geometric intuition behind equation (9). Equation (9) is closely related to the Lagrange multiplier formulation for constrained minimization (hence our choice of λ as a symbol for the multipliers of ∇G).

For the case of contact between a polyhedron A and a curved surface B, equation (9) is modified depending on whether p_c is coincident with a face, edge or vertex of the polyhedron. If p_c lies in a face, then F is the implicit function of the plane embedding the face. Otherwise, if p_c lies on an edge and $\nabla G(p_b,t_0)$ is not perpendicular to any adjoining face, p_a is constrained to be the extremal point on the edge. If p_c lies on a vertex, and $\nabla G(p_b,t_0)$ is not perpendicular to any adjoining edge (and hence any adjoining face), p_a is defined to be coincident with the vertex. In the above two cases, a system of equations similar to (9) is formed and used in place of equation (9).

Conditions E_1 thru E_4 are used to derive expressions for the derivatives of p_a and p_b. (We will consider numerical solutions of equation (9) to find p_a and p_b when we deal with collision determination in section 7.) Given extremal points p_a and p_b, we can find $\dot{p}_a$ and $\dot{p}_b$ by making use of the implicit function theorem for simultaneous equations from calculus[16]. This theorem asserts that under proper conditions, p_a and p_b may be regarded as functions of time; the theorem also gives an analytic expression for the derivatives of p_a and p_b (with respect to time). We will write the Jacobian determinant of a set of vector functions $H_1(\vec{x})$ thru $H_n(\vec{x})$ as

$$\frac{\partial(H_1,\cdots,H_n)}{\partial(x_1,\cdots,x_n)} = \begin{vmatrix} \frac{\partial H_1}{\partial x_1} & \cdots & \frac{\partial H_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial H_n}{\partial x_1} & \cdots & \frac{\partial H_n}{\partial x_n} \end{vmatrix}. \tag{10}$$

If we are dealing with 3-space, then p_a and p_b are each three (scalar) functions of time: $p_{a_x}(t)$, $p_{a_y}(t)$ and $p_{a_z}(t)$ and similarly for p_b. The implicit function theorem, applied to equation (9) yields

$$\dot{p}_{a_x}(t) = -\frac{K}{J} \tag{11}$$

where

$$K = \frac{\partial(E_1,E_2,E_3,E_4)}{\partial(t,p_{a_y},p_{a_z},p_{b_x},p_{b_y},p_{b_z},\lambda_1,\lambda_2)} \tag{12}$$

and

$$J = \frac{\partial(E_1,E_2,E_3,E_4)}{\partial(p_{a_x},p_{a_y},p_{a_z},p_{b_x},p_{b_y},p_{b_z},\lambda_1,\lambda_2)}. \tag{13}$$

Similar expressions give the derivatives for p_{a_y}, p_{a_z}, p_{b_x}, p_{b_y} and p_{b_z}. Appendix E discusses possible ill-conditioning of the Jacobian matrix J. If we are given the extremal points p_a and p_b, λ_1 and λ_2 are easily determined and $\dot{p}_a$ and $\dot{p}_b$ are easily calculated (assuming the needed derivatives of F and G are at hand). However, an expression for $\ddot{\chi}$ involves the symbolic computation of $\ddot{p}_a$ and $\ddot{p}_b$; these in turn require derivatives of K and J. Unfortunately, a symbolic expression for the determinant of K or J is impractical. Although K and J have considerable block structure, block structure cannot be exploited in computing determinants. In its present form, the Jacobian determinant contains more than 1,000 terms of seven factors each; the derivative would contain far more terms. A transformation of coordinate systems and functions is presented in the next section that yields more tractable expressions.

6. Coordinate Transformations

The formulation for $\dot{p}_a$ and $\dot{p}_b$ in the last section involved derivatives of determinants of 8×8 matrices. By choosing an appropriate coordinate system and transforming the representation of the surface functions F and G, we can find a tractable representation for $\dot{p}_a$ and $\dot{p}_b$.

First, we assume a rotated coordinate system in which $\nabla F(p_a,t_0)$ and $\nabla G(p_b,t_0)$ are colinear with the z axis at time t_0, with $\nabla G(p_b,t_0)$ directed positively along z. (We will employ the standard right-handed coordinate system used to depict functions $z=h(x,y)$, with z the vertical axis). The effect of this rotation on derivatives of F and G is discussed in appendix B. Next, we explicitly model A near p_a as a time-varying scalar function f of x and y. Where F was a function $F(x,y,z,t)$, f is a function $f(x,y,t)$. The justification for the existence of f is the implicit function theorem of calculus. The formal definition of f is

$$F(x,y,f(x,y,t),t) = 0. \tag{14}$$

Thus, the point $(x,y,f(x,y,t))$ is a point on F at time t. A function g is chosen similarly for G. In this new coordinate system, the extremal points p_a and p_b share the same x and y coordinates at time t_0. At times t near t_0, A and B do not penetrate as long as the z value of p_a is greater or equal to the z value of p_b (figure 8). This can be expressed as

$$\chi(t) = f(p_{a_x},p_{a_y},t) - g(p_{b_x},p_{b_y},t). \tag{15}$$

The condition $\ddot{\chi} \geq 0$ is simply

$$\frac{d^2}{dt^2} f(p_{a_x},p_{a_y},t_0) - \frac{d^2}{dt^2} g(p_{b_x},p_{b_y},t_0) \geq 0. \tag{16}$$

The extremal points are then expressed in terms of f and g, without the use of the multipliers λ_1 and λ_2 (see appendix C, equation (42)). This change of coordinate systems and functions reduces the number of variables from 8 to 4; this allows a formulation in terms of 4×4 matrix determinants as opposed to the 8×8 matrices of the previous section. While this is an improvement, calculating the derivatives of the determinants constructed from equation (42) is still a formidable challenge; however, they are no longer needed. The real advantage of the new formulation is that the second derivatives of p_{a_x}, p_{a_y}, p_{b_x} and p_{b_y} are *not* required when computing $\ddot{\chi}$. Equation (47) of appendix C shows how the second derivatives of p_a and p_b drop out of the expression for $\frac{d^2}{dt^2}f$ and $\frac{d^2}{dt^2}g$. The final result is a fairly simple symbolic expression for $\ddot{\chi}$.

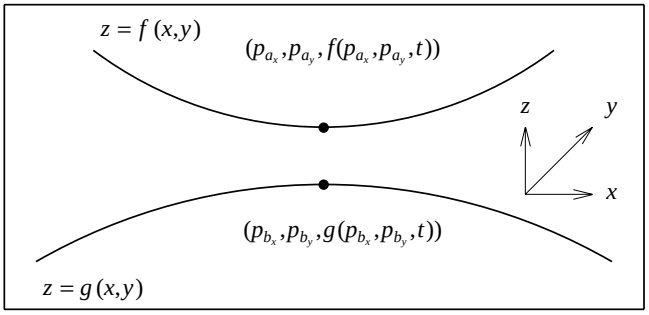

Figure 8. Side view of the implicit surfaces F and G expressed explicitly by f and g.

7. Contact Determination

Given the vast literature on collision detection in the computational geometry and robotics fields, it is with some trepidation that we present new algorithms for collision detection and determination of contact points. The algorithms presented are very basic; nonetheless we have found them to be extremely efficient for our simulations. In a dynamic simulation, a series of collision detection problems is encountered. Each problem is similar to the collision detection problem posed and solved during the previous time step. The focus of the algorithms presented in this section is using information from the previous time step to solve the collision detection problem for the current time step. The collision detection problems addressed in computational geometry and robotics are of a different nature.

In the field of computational geometry, collision detection algorithms are by and large restricted to static geometrical configurations. Algorithms are typically developed to solve a single instance of a problem involving fixed objects in the smallest asymptotic time complexity. The use of algorithms of this nature for collision detection during simulation essentially ignores any information discovered in previous time steps. Gilbert *et al.*[8] present an algorithm that efficiently computes the minimum distance between convex objects. Additionally, the algorithm can use information from previous time steps for fast initialization, and would appear to be an attractive candidate for detecting collisions. However, we are not interested in the value of the minimum distance *per se.* and we feel that the algorithms presented below are better suited to our simulation environment. For the simpler problem of determining the disjointness of two convex polyhedra with a total of n vertices, algorithms with asymptotic time complexities of $O(n\log n)$ and even $O(n)$ have been achieved; however, it is not clear that these algorithms are practically useful for reasonable values of n[8].

In the field of robotics, algorithms are developed for geometric problems involving objects with specified continuous motions over some time period; for example, determining the first

collision between polyhedral objects[4]. These algorithms presuppose known paths for the objects and detect collisions over the length of the paths. In our case, the paths of the objects are not known.

We have found that the geometric relationship between objects in our simulations does not change very much between successive time steps. As a result, collision detection algorithms for our simulation environment should be structured to take maximum advantage of the geometric coherence between successive time steps. Our simulator uses contact determination algorithms that assume a high degree of coherence between time steps. If the relative displacements of objects between successive time steps are large, this assumption breaks down. However, our experience has been that numerical considerations in solving the differential equations of motion limit the size of the time step. We choose not to treat the case where an object passes completely through another object in one time step; one solution to this problem is 4D space-time swept volume algorithms to detect collisions[13].

In this paper we limit our objects to the union of convex polyhedra and strictly convex closed surfaces. We will refer to these polyhedra and curved surfaces as *primitives*. The first geometric problem we consider is the pairwise comparison of primitives to determine inter-penetration.[4] Our primary mechanism for exploiting coherence will be through the use of *witnesses* to the decision problem of inter-penetration. A witness is some piece of information that can be used to quickly answer a decision problem. We will utilize coherence by caching witnesses from one time step to the next; hopefully a witness from the previous time step will be a witness during the current time step.

Since primitives are convex, a pair of primitives do not inter-penetrate if and only if a separating plane between them exists.[5] A separating plane between two objects is a plane such that each object lies in a different half-space of the plane. A given plane can be verified to be a separating plane between two convex polyhedra in time $O(n)$ where n is the total number of vertices of the two polyhedra. Cundall[5] performed pairwise collision detection during simulation by initially finding separating planes that were approximately equidistant from each polyhedron of the pair. At later time steps, numerical techniques were used to quickly update the separating planes so that they maintained their equidistant relationship between pairs of polyhedra. A simpler solution exists however; for disjoint or contacting convex polyhedra, it can be shown that a separating plane exists which either embeds at least one face of one of the polyhedra or embeds an edge from each polyhedron. If a pair of polyhedra are shown not to inter-penetrate by one of these separating planes, we cache the face or two edges embedded in the separating plane as a witness. At the next time step, we use the cached face or edges to form a new separating plane. In this manner the new separating plane is obtained simply from the old one without the need for any numerical computations. Even better, by caching the nearest face or edge to the separating plane from each polyhedron, disjointness can usually be verified in sublinear time (figure 9).

If two polyhedra are inter-penetrating, it is almost always the case that either a vertex of one polyhedron is inside the other, or an edge of one polyhedron has intersected a face of the other. In this case, the inter-penetrating vertex, or intersecting edge and face are cached as a witness to the inter-penetration. In subsequent comparisons, the witness is used to quickly check for inter-penetration. When it is necessary to initially find a witness, a sophisticated computational geometry algorithm such as Gilbert *et al.*[8] might be employed. Currently we use exhaustive search to initially find a witness; we have found the added expense, amortized over the length of the simulation, to be negligible.

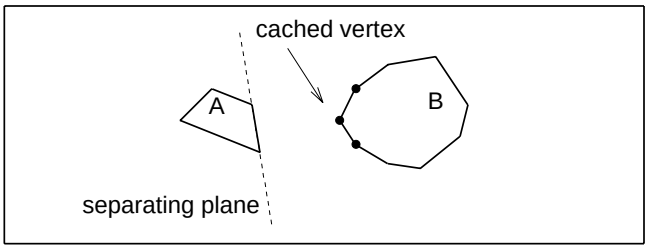

Figure 9. Sublinear time verification of disjointness based on cached witnesses.

The use of separating planes also makes the contact determination step simple. Contact points between a pair of polyhedra separated by a plane P can only occur on the plane P. Given the separating plane, the contact points are quickly and efficiently determined by comparing only those faces, edges or vertices coincident with the separating plane. This determination can itself be performed quickly by caching information from the previous time step.

For comparing two curved surfaces, we must employ numerical techniques to determine disjointness. We make use of the concept of extremal points from section 5 in determining disjointness. Given two disjoint convex curved surfaces, the extremal points are the points of minimum distance and are a witness to the disjointness of the surfaces. Otherwise, the extremal points near the intersection of the surfaces are a witness that the surfaces do inter-penetrate (figure 6). To find these extremal points, a non-linear equation solver may be used to solve equation (9) for p_a and p_b[6]; see Forsythe *et al.*[7]. However, equation (9) admits multiple solutions of p_a and p_b; we are interested in the solution that globally minimizes $\|p_a - p_b\|$.[7] Non-linear equation solvers proceed from some initial estimate of the solution to an exact solution (within numerical tolerances). If we are initially finding the extremal points between the surfaces, we require some rough estimate of p_a and p_b, so that the solver will converge to the proper solution. For implicit surfaces, we can initially estimate p_a and p_b by intersecting the two surfaces with the line connecting the centroids of the two surfaces. For parametric surfaces, we can generate and store a coarse mesh of surface points for each surface and use the parametric coordinates of the minimum distance pair of points as an initial estimate. Using the initial estimate as a starting point, the solver converges to the proper solution of the extremal points.

Once the extremal points p_a and p_b are determined, they are cached for the next time step. In subsequent time steps, the cached extremal points are used as an initial estimate and the solver converges in a few iterations to the new extremal points. Furthermore, the accuracy of this initial estimate can be significantly improved as follows. If in addition to caching the extremal points p_a and p_b at time t_0 we cache $\dot{p}_a$ and $\dot{p}_b$, we can estimate p_a and p_b at time $t_0 + \Delta t$ by

[4] Initial techniques such as hierarchical bounding volumes and spatial subdivision can be used to limit the number of pairs of primitives considered by this step. This initial step also benefits greatly by exploiting coherence.

[5] Note that convexity is crucial as this argument does not hold for concave objects. The collision detection problem for concave objects is a considerably more difficult problem than for convex objects.

[6] For parametrically defined surfaces, we replace equation (9) with a non-linear equation that defines the extremal points in terms of parametric coordinates; see appendix D for details.

[7] The points p_a and p_b that *maximize* the distance between the surfaces satisfy equation (9), but they are not the solution we are interested in. Parametric surface have multiple solutions of equation (56) for p_a and p_b in terms of their parametric coordinates.

$$p_a(t_0 + \Delta t) = p_a(t_0) + \Delta t\, \dot{p}_a(t_0) \tag{17}$$

and similarly for p_b. (See appendix D for the parametric case). Improving the accuracy of the initial estimates of p_a and p_b adds to the speed and robustness of the algorithm. The derivatives of $\dot{p}_a(t_0)$ and $\dot{p}_b(t_0)$ can be calculated in terms of determinants of 4×4 matrices as described in appendix C. Once two curved surfaces have been found not to inter-penetrate, the contact determination process consists of merely comparing the distance between the extremal points to some numerical threshold.

Comparisons between a polyhedral primitive and a curved surface primitive are handled in an analogous manner to contact between a polyhedron and a curved surface (section 5).

8. Determining the Collision Time

The last problem in the collision detection phase is backtracking to the point of a collision (step E of figure 1). During a simulation, it may happen that two objects *A* and *B* come into colliding contact at some time t_c. Suppose that t_0 was the time of the previous time step when the objects had not yet collided, and at time t_1 ($t_0 < t_c < t_1$) it is found that *A* and *B* have inter-penetrated. When this occurs, the simulator makes a prediction t_p of the time t_c at which the initial collision occurred, backs up to time t_0 and then moves forward to time t_p. If the simulator finds that *A* and *B* have not yet collided at t_p, it assumes that $t_p < t_c$ and makes a larger prediction for t_c. Conversely, if the simulator finds that *A* and *B* have inter-penetrated at t_p, it assumes that $t_c < t_p$, and makes a smaller prediction for t_c. Otherwise, the simulator has found t_c to within numerically accepted tolerances and may proceed.

Conceptually, this can be viewed as a root finding problem. Previous papers have solved this root finding problem by using bisection[13] or *regula falsa*[1]. The bisection method is extremely robust, simple to implement, and independent of geometry; however, it has relatively slow convergence, especially where great accuracy is required. The *regula falsa* method linearly interpolates the distance between *A* and *B* at t_0 and the amount of inter-penetration at t_1 to predict t_c. The *regula falsa* method handles any geometry as long as measures of separation and inter-penetration are available; additionally, the method converges faster than bisection. *regula falsa* is not as robust as bisection but a hybrid bisection-*regula falsa* algorithm[7] works well in practice.

An alternative to *regula falsa* is Newton's method. Newton's method requires both a measure of separation between *A* and *B* at t_0 and the relative approach velocity at time t_0. These quantities however are exactly modeled by χ and $\dot{\chi}$. Newton's method for solving $h(t) = 0$ near the point t_0 is based on the Taylor series expansion

$$h(t) = h(t_0) + \sum_{n=1}^{\infty} \frac{h^{(n)}(t_0)}{n!}(t - t_0)^n. \tag{18}$$

By throwing out terms for $n > 1$ and replacing h with χ, we obtain the linear approximation

$$t = t_0 - \frac{\chi(t_0)}{\dot{\chi}(t_0)}. \tag{19}$$

While Newton's method gives better convergence than the *regula falsa* method, both *regula falsa* and Newton's method consistently either under-estimate or over-estimate t_c for constant (non-zero) acceleration. Since constant acceleration occurs frequently, it makes sense to predict t_c by using a quadratic model as opposed to the linear model used by Newton's method. The quadratic model requires χ, $\dot{\chi}$ and $\ddot{\chi}$; section 5 and appendix C show how χ, $\dot{\chi}$ and $\ddot{\chi}$ are calculated for polyhedral and curved objects. Following the derivation of Newton's algorithm, we predict t_c by

$$\chi(t) = \chi(t_0) + \dot{\chi}(t_0)(t - t_0) + \frac{\ddot{\chi}(t_0)}{2}(t - t_0)^2. \tag{20}$$

Solving for t we obtain

$$t = t_0 + \frac{-\dot{\chi}(t_0) \pm \sqrt{\dot{\chi}(t_0)^2 - 2\ddot{\chi}(t_0)\chi(t_0)}}{\ddot{\chi}(t_0)}. \tag{21}$$

We set t_p to be the smallest real root of equation (21) greater than t_0; if no such root exists, or $\ddot{\chi}(t_0)$ is zero (within numerical tolerance), we use Newton's method. The method is made robust by incorporating a bisection step whenever convergence is slow[7]. For constant acceleration, equation (21) gives an exact result as long as χ is a linear measure of the distance.[8] For non-constant acceleration, the quadratic model still converges faster than Newton's method, close to t_c.

9. Conclusion

Table 1 gives a rough indication of the running time of two simulations (figures 10 and 11) on a Hewlett Packard 835 workstation. A "cache miss" means that a new witness was computed from scratch, while a "cache hit" means that a previously cached witness was successfully updated to a witness for the current time step. The first simulation had 97 polygons and 6 curved surfaces and encountered 60 discontinuities while time stepping. The second simulation had 89 polygons and 102 curved surfaces and encountered 343 discontinuities while time stepping.

Figure	Total No. Time Steps	Cache Hits	Cache Misses	CPU Minutes
10. Jack	1,475	5,243	42	2.1
11. Dice	4,162	345,793	1,384	78.6

Table 1. Running times and caching effectiveness.

Acknowledgements

This research was funded by an AT&T Bell Laboratories PhD Fellowship and two NSF grants (#DCR8203979 and #ASC8715478). Simulations were performed on equipment generously donated by the Hewlett Packard Corporation. Some displays were computed using an AT&T Pixel Machine, donated by AT&T.

Appendix A: Rigid Body Motion of Surfaces

The formulation for the constraint function χ between curved surfaces models the surfaces as time-varying implicit functions $F(p,t)$, $F : R^3 \times R \to R$. A point p is on the surface of F at time t if $F(p,t) = 0$. We will represent the shape $F(p,t)$ in terms of a rest function $F_r(p)$ and a rigid body transformation $T(p,t)$.

Let $F_r(p)$ be a time-invariant function from R^3 to R; F_r defines a rest shape by the equation $F_r(p) = 0$. Let a rigid body motion be defined by the affine transformation $T : R^3 \times R \to R^3$ by

$$T(p,t) = c(t) + R(t)p \tag{22}$$

where $R(t)$ is a 3×3 rotation matrix and $c(t)$ is a point in R^3. Define $\tilde{T}$ as

$$\tilde{T}(p,t) = R^T(t)(p - c(t)) \tag{23}$$

[8] For polyhedral contact, χ of equation (3) is a linear measure of distance, and equation (21) converges in one step to t_c for constant accelerations. For curved surfaces, χ of equation (8) is weighted by $\|\nabla G\|$. For reasonably scaled functions, $\|\nabla G\|$ does not vary much over the range of the prediction.

so that

$$\tilde{T}(T(p,t),t) = T(\tilde{T}(p,t),t) = p. \tag{24}$$

If the linear and angular velocities of the rigid body motion T at time t are $v(t)$ and $\omega(t)$ then

$$\dot{c}(t) = v(t), \quad \dot{R}(t) = \omega^*(t)R(t) \tag{25}$$

where $\omega^*(t)$ is the dual[3, 9] of $\omega(t)$.[9] Define the point-velocity function $V(p,t)$ as

$$V(p,t) = v(t) + \omega(t)\times(p - c(t)). \tag{26}$$

Then

$$\begin{aligned}\dot{\tilde{T}}(p,t) &= \left(\omega^*(t)R(t)\right)^T(p - c(t)) - R(t)^T v(t)\\ &= -R(t)^T\left(\omega(t)\times(p-c(t)) + v(t)\right) = -R(t)^T V(p,t).\end{aligned} \tag{27}$$

We can represent the movement of F_r by the rigid body motion $T(p,t)$ by defining

$$F(p,t) = F_r(\tilde{T}(p,t)). \tag{28}$$

If q is some point on F_r then at time t the point $T(q,t)$ is a point of F since

$$F(T(q,t),t) = F_r(\tilde{T}(T(q,t),t)) = F_r(q) = 0. \tag{29}$$

Using the above definitions and the relations $\nabla F(p,t) = R(t)\nabla F_r(T(p,t))$ (see appendix B) and ${F_r}'^T = \nabla F_r$,

$$\begin{aligned}\dot{F}(p,t) &= F_r'(\tilde{T}(p,t))\dot{\tilde{T}}(p,t) = F_r'(\tilde{T}(p,t))(-R(t)^T V(p,t))\\ &= -\left(R(t)F_r'(\tilde{T}(p,t))^T\right)^T V(p,t) = -\nabla F(p,t)\cdot V(p,t).\end{aligned} \tag{30}$$

By differentiating equation (30) with respect to p and using $\nabla V(p,t) = \omega^*(t)$

$$\begin{aligned}\dot{\nabla F}(p,t) &= -F''(p,t)V(p,t) - \nabla F(p,t)\cdot\nabla V(p,t)\\ &= -F''(p,t)V(p,t) - \nabla V(p,t)^T\nabla F(p,t)\\ &= -F''(p,t)V(p,t) - \omega^*(t)^T\nabla F(p,t)\\ &= -F''(p,t)V(p,t) + \omega(t)\times\nabla F(p,t).\end{aligned} \tag{31}$$

Differentiating equation (30) with respect to time,

$$\ddot{F}(p,t) = -\left(\dot{\nabla F}(p,t)\cdot V(p,t) + \nabla F(p,t)\cdot\dot{V}(p,t)\right). \tag{32}$$

Since $V(p,t)$ is the point velocity of p in its rigid body frame, $\dot{V}(p,t)$ is the point acceleration of p and is a linear function of the forces in the system. Thus, $\ddot{F}$ is also a linear function of force.

Appendix B: Rotation of Coordinate Systems

In section 6, the coordinate axes were rotated so that $\nabla G(p_a,t_0)$ would be colinear with $\hat{z}$, the unit z axis vector. Let R be the change of basis matrix; R is a rotation matrix such that

$$R\nabla G(p_a,t_0) = \|\nabla G(p_a,t_0)\|\hat{z}. \tag{33}$$

Note that R is constant with respect to time. Let ∇G_0 and G_0'' be the derivatives of G in the original coordinate system. In the rotated coordinate system, the derivatives become

$$\nabla G = R\nabla G_0, \quad \dot{\nabla G} = R\nabla\dot{G}_0 \quad \text{and} \quad G'' = RG_0''R^T. \tag{34}$$

The derivatives $\dot{G}$ and $\ddot{G}$ are invariant under rotation. See Goldstein[9] for further discussion.

[9] Given a vector $a \in R^3$, a^* is the 3×3 (anti-symmetric) matrix such that for any vector $b \in R^3$, $a^*b = a\times b$.

Appendix C: Derivation of $\ddot{\chi}$

In section 6, a change of functions is introduced by writing the implicit functions F and G in terms of explicit functions f and g near the extremal points p_a and p_b. This change of functions is made in a coordinate system where both $\nabla F(p_a,t_0)$ and $\nabla G(p_b,t_0)$ are colinear with the z axis. The explicit functions f and g are defined by

$$F(x,y,f(x,y,t),t) = 0 \quad \text{and} \quad G(x,y,g(x,y,t),t) = 0. \tag{35}$$

The existence of f and g is seen by the implicit function theorem. By differentiating equation (35) we obtain

$$\frac{\partial f}{\partial x} = -\frac{\frac{\partial F}{\partial x}}{\frac{\partial F}{\partial z}}, \quad \frac{\partial f}{\partial y} = -\frac{\frac{\partial F}{\partial y}}{\frac{\partial F}{\partial z}} \quad \text{and} \quad \frac{\partial f}{\partial t} = -\frac{\frac{\partial F}{\partial t}}{\frac{\partial F}{\partial z}} \tag{36}$$

and similarly for g. Second derivatives of f and g are obtained by repeated differentiation of equation (35).

Using f and g, condition E_4 of equation (9) may be written componentwise as

$$\begin{pmatrix} p_{b_x}\\ p_{b_y}\\ g(p_{b_x},p_{b_y},t)\end{pmatrix} - \begin{pmatrix} p_{a_x}\\ p_{a_y}\\ f(p_{a_x},p_{a_y},t)\end{pmatrix} + \lambda_1\begin{pmatrix}\frac{\partial G}{\partial x}(p_b,t)\\ \frac{\partial G}{\partial y}(p_b,t)\\ \frac{\partial G}{\partial z}(p_b,t)\end{pmatrix} = \begin{pmatrix}0\\0\\0\end{pmatrix} \tag{37}$$

from which we obtain

$$\lambda_1 = \frac{f(p_{a_y},p_{a_y},t) - g(p_{b_x},p_{b_y},t)}{\frac{\partial G}{\partial z}(p_b,t)}. \tag{38}$$

Multiplying equation (37) by -1 and using equation (36) allows us to rewrite E_4 as

$$\begin{pmatrix}p_{a_x} - p_{b_x}\\ p_{a_y} - p_{b_y}\end{pmatrix} + (f(p_{a_x},p_{a_y},t) - g(p_{b_x},p_{b_y},t))\begin{pmatrix}\frac{\partial g}{\partial x}(p_{b_x},p_{b_y},t)\\ \frac{\partial g}{\partial y}(p_{b_x},p_{b_y},t)\end{pmatrix} = \vec{0}. \tag{39}$$

This new condition has one less equation than E_4 because of the reduction of variables from F to f. In a similar fashion, λ_2 of condition E_1 is eliminated and condition E_1 is rewritten as

$$\begin{pmatrix}\frac{\partial f}{\partial x}(p_{a_x},p_{a_y},t)\\ \frac{\partial f}{\partial y}(p_{a_x},p_{a_y},t)\end{pmatrix} - \begin{pmatrix}\frac{\partial g}{\partial x}(p_{b_x},p_{b_y},t)\\ \frac{\partial g}{\partial y}(p_{b_x},p_{b_y},t)\end{pmatrix} = \vec{0}. \tag{40}$$

Conditions E_2 and E_3 are no longer required since f and g give explicit definitions of the surfaces. Using the notation

$$\nabla f^T = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = f' \tag{41}$$

p_{a_x}, p_{a_y}, p_{b_x} and p_{b_y} may be defined as the solution to

$$\begin{cases} D_1\colon \nabla f(p_{a_x},p_{a_y},t) - \nabla g(p_{b_x},p_{b_y},t) = \vec{0}\\ D_2\colon \begin{pmatrix}p_{a_x} - p_{b_x}\\ p_{a_y} - p_{b_y}\end{pmatrix} + (f(p_{a_x},p_{a_y},t) - g(p_{b_x},p_{b_y},t))\nabla g(p_{b_x},p_{b_y},t) = \vec{0}.\end{cases} \tag{42}$$

The implicit function theorem for simultaneous equations gives the result

$$\dot{p}_{a_x} = -\frac{\frac{\partial(D_1,D_2)}{\partial(t,p_{a_y},p_{b_x},p_{b_y})}}{J}, \quad \dot{p}_{a_y} = -\frac{\frac{\partial(D_1,D_2)}{\partial(p_{a_x},t,p_{b_x},p_{b_y})}}{J} \tag{43}$$

(and similarly for $\dot{p}_{b_x}$ and $\dot{p}_{b_y}$) where

$$J = \frac{\partial(D_1,D_2)}{\partial(p_{a_x},p_{a_y},p_{b_x},p_{b_y})}. \tag{44}$$

The derivatives of p_{a_z} and p_{b_z} are simply $\dot{f}(p_{a_x},p_{a_y},t)$ and $\dot{g}(p_{b_x},p_{b_y},t)$.

Since D_1 and D_2 do not involve p_{a_z} and p_{b_z}, let p_a and p_b denote (only for the remainder of this appendix) the column vectors $(p_{a_x},p_{a_y})^T$ and $(p_{b_x},p_{b_y})^T$ at time t. Similarly, let $\dot{p}_a = (\dot{p}_{a_x},\dot{p}_{a_y})^T$ and $\dot{p}_b = (\dot{p}_{b_x},\dot{p}_{b_y})^T$ at time t. In section 6, we derived

$$\chi(t) = f(p_a,t) - g(p_b,t). \tag{45}$$

To obtain $\ddot{\chi}$, we must doubly differentiate f and g with respect to time. Using the chain rule,

$$\frac{d}{dt} f(p_a,t) = f'(p_a,t)\dot{p}_a + \dot{f}(p_a,t) = \nabla f(p_a,t)^T \dot{p}_a + \dot{f}(p_a,t). \tag{46}$$

Then

$$\frac{d^2}{dt^2} f(p_a,t) = \left(f''(p_a,t)\dot{p}_a + \dot{\nabla} f(p_a,t)\right)^T \dot{p}_a + \nabla f(p_a,t)^T \ddot{p}_a + \dot{f}'(p_a,t)\dot{p}_a + \ddot{f}(p_a,t) \tag{47}$$

$$= \dot{p}_a{}^T f''(p_a,t)\dot{p}_a + 2\dot{\nabla} f(p_a,t)^T \dot{p}_a + \nabla f(p_a,t)^T \ddot{p}_a + \ddot{f}(p_a,t).$$

From equation (36) and the fact that $\nabla F(p_a,t_0)$ is colinear with the z axis,

$$\nabla f(p_a,t_0)^T = (0,\ 0)^T. \tag{48}$$

This yields

$$\frac{d^2}{dt^2} f(p_a,t_0) = \dot{p}_a{}^T f''(p_a,t_0)\dot{p}_a + 2\dot{\nabla} f(p_a,t_0)^T \dot{p}_a + \ddot{f}(p_a,t_0). \tag{49}$$

A similar expression is obtained for g. Thus, neither $\ddot{p}_a$ nor $\ddot{p}_b$ is required for the symbolic computation of $\dddot{\chi}(t_0)$. Equation (49) is a linear function of $\dot{F}$ by its last term, $\dot{f}(p_a,t_0)$ which is in turn a linear function of the contact forces. Thus, $\dddot{\chi}(t_0)$ is itself a linear function of the contact forces.

Appendix D: Parametrically Defined Surfaces

Appendix C derived $\ddot{\chi}$ by defining explicit surfaces f and g that modeled the implicit surfaces F and G near extremal points. $\ddot{\chi}$ was then written in terms of the derivatives of the extremal points and the derivatives of f and g; in turn, the derivatives of f and g were expressed in terms of the implicit functions F and G. Given two parametric surfaces, S and T, the same thing can be done. We will first show how to express the derivatives of f and g in terms of the parametric functions S and T. We will then show how the extremal points and their derivatives are defined for parametric functions. This enables $\dddot{\chi}$ to be computed as shown in appendix C.

Let a time-dependent parametric function $S(u,v,t)$, $S: R^2 \times R \to R^3$, define a surface. We will write S in terms of three component functions S_x, S_y and S_z as

$$S(u,v,t) = \left(S_x(u,v,t),\ S_y(u,v,t),\ S_z(u,v,t)\right)^T. \tag{50}$$

As in appendix A, S is assumed to be the rigid body motion of some time-invariant base shape S_0;

$$S(u,v,t) = R(t)S_0(u,v) + c(t). \tag{51}$$

Differentiating with respect to t,

$$\begin{aligned}\frac{\partial S}{\partial t}(u,v,t) &= \omega^*(t)R(t)S_0(u,v) + v(t) \\ &= \omega^*(t)\Big(R(t)S_0(u,v) + c(t) - c(t)\Big) + v(t) \\ &= \omega \times \Big(S(u,v,t) - c(t)\Big) + v(t).\end{aligned} \tag{52}$$

Assume that at time t_0, the surface normal of S at the point $S(u_0,v_0,t_0)$ is colinear with the z axis. As in appendix C, we let $f(x,y,t)$ explicitly describe S near $S(u_0,v_0,t_0)$. The definition of f is

$$f(S_x(u,v,t),S_y(u,v,t),t) = S_z(u,v,t). \tag{53}$$

The existence of f follows from the implicit function theorem. Differentiating equation (53) with respect to u, v, and t, we obtain the system

$$\begin{aligned}\frac{\partial f}{\partial x}\frac{\partial S_x}{\partial u} + \frac{\partial f}{\partial y}\frac{\partial S_y}{\partial u} &= \frac{\partial S_z}{\partial u} \\ \frac{\partial f}{\partial x}\frac{\partial S_x}{\partial v} + \frac{\partial f}{\partial y}\frac{\partial S_y}{\partial v} &= \frac{\partial S_z}{\partial v} \\ \frac{\partial f}{\partial x}\frac{\partial S_x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial S_y}{\partial t} + \frac{\partial f}{\partial t} &= \frac{\partial S_z}{\partial t}\end{aligned}. \tag{54}$$

Thus, the first partials of f may be expressed in terms of partial derivatives of the parametric function S by solving a simple linear system. The second partials of f are obtained by differentiating the system of equations (54) and solving another linear system. A normal vector $S_N(u_0,v_0,t_0)$ to $S(u_0,v_0,t_0)$ at time t_0 is given by

$$S_N(u_0,v_0,t_0) = \frac{\partial S}{\partial u}(u_0,v_0,t_0) \times \frac{\partial S}{\partial v}(u_0,v_0,t_0). \tag{55}$$

Suppose that $T(u,v,t)$ is the parametric function for a second surface, with T_N defined accordingly. Let the parametric coordinates of the extremal points p_a and p_b (where p_a is on S and p_b is on T) be (u_a,v_a) and (u_b,v_b). The analogue to equation (9) is then

$$\begin{cases} E_1\colon\ S_N(u_a,v_a,t) + \lambda_2 T_N(u_b,v_b,t) = \vec{0} \\ E_2\colon\ (T(u_b,v_b,t) - S(u_a,v_a,t)) + \lambda_1 T_N(u_b,v_b,t) = \vec{0}. \end{cases} \tag{56}$$

As in section 5, we may regard u_a, v_a, u_b and v_b as functions of time; derivatives are then given by

$$\dot{u}_a = -\frac{\frac{\partial(E_1,E_2)}{\partial(t,v_a,u_b,v_b)}}{J}, \quad \dot{v}_a = -\frac{\frac{\partial(E_1,E_2)}{\partial(u_a,t,u_b,v_b)}}{J} \tag{57}$$

(and similarly for $\dot{u}_b$ and $\dot{v}_b$) where

$$J = \frac{\partial(E_1,E_2)}{\partial(u_a,v_a,u_b,v_b)}. \tag{58}$$

Given the derivatives of the parametric coordinates of the extremal points, we can calculate the needed derivatives of p_{a_x}, p_{a_y}, p_{b_x} and p_{b_y} (for equation (49)). From equation (50) and since $p_a = S(u_a,v_a,t)$,

$$\dot{p}_{a_x} = \frac{\partial S_x(u_a,v_a,t)}{\partial u}\dot{u}_a + \frac{\partial S_x(u_a,v_a,t)}{\partial v}\dot{v}_a + \frac{\partial S_x(u_a,v_a,t)}{\partial t} \tag{59}$$

and similarly for $\dot{p}_{a_y}$, $\dot{p}_{b_x}$ and $\dot{p}_{b_y}$.

For contact determination, equation (56) is solved (in place of equation (9)) to find (u_a,v_a) and (u_b,v_b). The extremal points p_a and p_b are then cached in terms of their parametric coordinates

(u_a, v_a) and (u_b, v_b). Equation (17) is replaced by

$$u_a(t_0 + \Delta t) = u_a(t_0) + \Delta t\, \dot{u_a}(t_0) \tag{60}$$

and similarly for v_a, u_b and v_b.

Appendix E: Ill-conditioned Jacobian Determinants

Given the formulation of $\dot{p}_a$ and $\dot{p}_b$ by equation (43) in appendix C, a natural concern arises over the possible ill-conditioning of the Jacobian matrix $\frac{\partial(D_1, D_2)}{\partial(p_{a_x}, p_{a_y}, p_{b_x}, p_{b_y})}$ (and similarly for the parametric case). An examination of the Jacobian shows that its condition depends on the Gaussian curvatures of *F* and *G* at the extremal points. As long as at least one of the two curvatures is non-zero the Jacobian determinant is non-zero. However, if both curvatures become zero or near zero at the extremal points, the Jacobian also becomes zero or near zero and $\dot{p}_a$ and $\dot{p}_b$ diverge. Thus, the overall derivation for forces between pairs of curved objects is meant to be applied to pairs where at least one of the objects really is curved at the contact point. In regions where both curvatures approach but do not attain zero, the contact force can change rapidly. If both curvatures achieve zero simultaneously, the contact force may be unbounded (although its integral is not). The former case can be handled (inefficiently) by taking sufficiently small time steps. The latter case would seem to require some sort of analytic integral of the contact force; this is a topic for further research.

References

1. Baraff, D., ''Analytical methods for dynamic simulation of non-penetrating rigid bodies,'' *Computer Graphics (Proc. SIGGRAPH)*, vol. 23, pp. 223-232, 1989.
2. Baraff, D., ''Determining frictional inconsistency for rigid bodies is NP-complete,'' *Technical Report TR 90-1112*, Department of Computer Science, Cornell University, 1990.
3. Barzel, R. and Barr, A.H., ''A modeling system based on dynamic constraints,'' *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, pp. 179-188, 1988.
4. Canny, J., ''Collision detection for moving polyhedra,'' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 200-209, 1986.
5. Cundall, P.A., ''Formulation of a three-dimensional distinct element model — Part I. A scheme to represent contacts in a system composed of many polyhedral blocks,'' *International Journal of Rock Mechanics, Mineral Science and Geomechanics*, vol. 25, no. 3, pp. 107-116, 1988.
6. Featherstone, R., *Robot Dynamics Algorithms*, Kluwer, Boston, 1987.
7. Forsythe, G.E., Malcolm, M.A., and Moler, C.B., *Computer Methods for Mathematical Computations*, Prentice Hall, Inc., Englewood Cliffs, 1977.
8. Gilbert, E.G., Johnson, D.W., and Keerthi, S.S., ''A fast procedure for computing the distance between complex objects in three space,'' *IEEE Journal of Robotics and Automation*, vol. 4, pp. 193-203, 1988.
9. Goldstein, H., *Classical Mechanics*, Addison-Wesley, Reading, 1983.
10. Goyal, S., ''Second order kinematic constraint between two bodies rolling, twisting and slipping against each other while maintaining point contact,'' *Technical Report TR 89-1043*, Department of Computer Science, Cornell University, 1989.
11. Lötstedt, P., ''Mechanical systems of rigid bodies subject to unilateral constraints,'' *SIAM Journal of Applied Mathematics*, vol. 42, no. 2, pp. 281-296, 1982.
12. Lötstedt, P., ''Numerical simulation of time-dependent contact friction problems in rigid body mechanics,'' *SIAM Journal of Scientific Statistical Computing*, vol. 5, no. 2, pp. 370-393, 1984.
13. Moore, M. and Wilhelms, J., ''Collision detection and response for computer animation,'' *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, pp. 289-298, 1988.
14. Murty, K.G., *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann Verlag, Berlin, 1988.
15. Neĭmark, Ju.I. and Fufaev, N.A., *Dynamics of Nonholonomic Systems*, American Mathematical Society, 1972.
16. Taylor, A.E. and Mann, R.M., *Advanced Calculus*, John Wiley & Sons, Inc., New York, 1983.
17. Tomlin, J.A., ''Robust implementation of Lemke's method for the linear complementarity problem,'' *Technical Report SOL 76-24*, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1976.

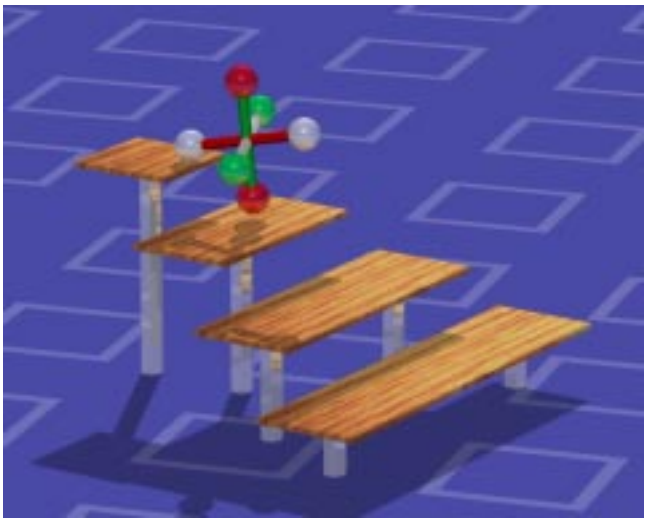

Figure 10. Falling jack.

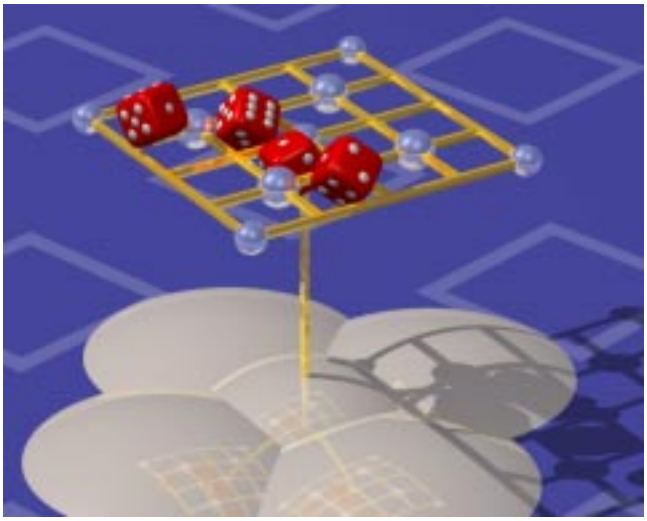

Figure 11. Falling dice.