Engineering with Computers (1994) 10: 175–195
© 1994 Springer-Verlag London Limited

Engineering with Computers

Simulation of Dynamics of Interacting Rigid Bodies Including Friction II: Software System Design and Implementation

Suresh Goyal, Elliot N. Pinson and Frank W. Sinden*
AT&T Bell Laboratories, Murray Hill, New Jersey 07974-0636, USA

Abstract. *This is the second of two papers that deal with the problem of modeling contact (impact, sliding, rolling) between unconstrained rigid bodies, including friction. In a companion paper* [1] *we showed that the main underlying problem concerns the ability to do efficient* contact mechanics *when bodies interact through impact and/or sustained contact. Contact mechanics involves two aspects: detection of contact between bodies and estimation of contact forces. These forces are complicated in character and difficult to estimate because they depend on the material response of the contacting objects, on the duration of contact (very short duration impact, or more sustained contact), frictional interaction at the surfaces, geometry of contact, etc. In* [1] *we proposed a conceptual model in which linear elastic (springs) and viscous (dampers) elements acting at points of contact between objects generate all contact forces. In this paper we describe how the contact model has been implemented in the software of a working computer simulation system. The major aspects of this process are: formulation of a discrete version of the contact model; calculation of model parameters to reflect material properties; geometric representation of objects (in our system, objects are modeled as convex polyhedra); algorithms to detect and evaluate contacts among objects (a process called contact analysis); and estimation and control of model response for stable numerical integration of equations of motion. A graphical user interface displays a three-dimensional (3-D) perspective animation of the solution using full color shaded surface images. While the simulation may not be accomplished in real time, solutions can be saved in files for real-time visualization.*

Keywords. Animation; Contact analysis; Contact detection; Convex polyhedra; Material parameters; Numerical integration

1. Introduction

We have developed a software system for simulating the dynamics of a system of freely interacting rigid bodies including frictional forces. Systems like these have wide ranging applications as a *software laboratory* for prototyping, automation, analysis, education, training and entertainment – issues that we have discussed in more detail in a companion paper [1]. We also showed in [1] that the main underlying problem concerns the ability to do efficient *contact mechanics* when bodies interact through impacts and/or sustained contact. From the point of view of simulation this involves two aspects: detection of contact between bodies and the estimation of appropriate contact forces. These forces are complicated in character and difficult to estimate because they depend on the material properties of the contacting objects, on the time duration of contact (very short duration impact or a more sustained contact), frictional interaction at the surfaces, geometry of contact, etc. In essence, material interaction during contact accomplishes transfer of momentum and dissipation of energy. There are two broad approaches to estimating contact forces: the soft contact approach that estimates forces by modeling localized deformation in the vicinity of the contact, and the traditional hard contact approach that calculates forces on the assumption that bodies are infinitely rigid and do not penetrate each other.

In [1] we described some issues pertaining to both of these approaches. We then proposed a contact model based on the soft contact approach that generates contact forces (including Coulomb-like dry friction) by modeling localized non-permanent material deformation of the contacting surfaces. In this paper we describe the implementation of our contact scheme in a software system.

Previous work on the dynamics of interacting rigid bodies owes much to efforts of late 19th century mechanicians such as Routh [2], and more detailed recent compilations [3]. The subject has recently received increased attention in the Computer Science community because of its relevance to the generation of realistic computer graphics animations. Hahn [4] has described a method using impact analysis and

Correspondence and offprint requests to: Dr Suresh Goyal, AT&T Bell Labs, 600 Mountain Avenue, Rm 1B-212, Murray Hill, NJ 07974-0636, USA.
* Authors are listed in alphabetical order

collision detection to obtain approximate solutions for rigid body interactions, and Armstrong and Green [5] apply simplified approximations to the animation of human figures. Baraff [6] has described an analytical method for calculating forces between systems of rigid bodies in resting (i.e. non-colliding) contact that requires solution of a quadratic programming problem or, if a heuristic approach is taken, a linear programming problem. Walton *et al.* [7] have reported two-dimensional applications in connection with the response of rock formations to explosions, and Cundall [8] and Cundall and Strack [9] have described work using soft contact models for rock mechanics applications.

Primarily, our software system demonstrates the feasibility of obtaining detailed and robust simulations of the motion of interacting rigid bodies by using a conceptually consistent spring–damper model for estimating contact forces. It does not have the combinatorial complexity associated with the quadratic program in the traditional approach, nor does it require different formulations for treating impacts and sustained contacts. Our contact model explicitly includes the effects of friction (which have not been treated adequately previously) and energy loss during impact, and contains parameters (coefficient of friction, spring and damper coefficients) that can be related to measurable properties of real materials and impact events (e.g. Young's modulus and the coefficient of restitution). We describe some experiments for choosing model parameters to produce behavior consistent with specific materials. Since we treat all contacts as having a finite duration and multiple simultaneous contacts are allowed, it is also possible to examine in detail the time-varying dynamic loads that occur during complex impacts. This is a very useful feature for purposes of analysis and mechanical design, which is not available with the traditional approach of modeling impacts singly through instantaneous momentum impulses.

Our simulation program consists of two independent modules that interact through well defined interfaces. They are:

1. The *dynamics module* that formulates the governing equations for the system, estimates the contact forces and other loads (forces and moments) based on information provided by the geometry module, and integrates the equations of motion to systematically update the state of the system.
2. The *geometry module* that models the shape, mass and inertial attributes of each body. The 'world' of objects is restricted to a subclass of convex polyhedra, which leads to simpler (and faster) geometry algorithms. The main geometric problem, called *contact analysis*, involves detecting the onset of a contact, evaluating details of the contact, tracking it through the entire contact episode and determining (with the aid of the dynamics module) the end of contact. Detecting initial contact is an exercise in *collision detection*, a subject that has been discussed extensively in the literature. We have our own method for solving this problem, but it is similar to previously described techniques. However, *contact analysis*, including the logic used to convert geometric interference information into a list of active contacts for the force model, involves substantially new issues. The geometry module also provides a display system that renders the results of the simulation in the form of an interactive animation on a computer display terminal.

Our paper describes this subject matter in several sections. Section 2 presents an explanation of the overall program structure. Section 3 describes the dynamics module, including sub-sections on the discrete contact model that we implement and its associated state equations, details about tuning model parameters and estimating and controlling model response for numerical stability. Section 4 presents the details of geometric processing, including object representation, the equation integration/geometric analysis paradigm, contact analysis and special cases that must be considered. Section 5 describes the simulation system environment, section 6 discusses performance issues and section 7 presents our conclusions.

2. Simplified Overall Program Structure

To put the different functions in our software system in proper perspective, a simplified flowchart is shown in Fig. 1. At program initiation, the geometry module reads a user-specified input file to create a world of convex polyhedra. The file contains object shape specifications, initial positions and velocities, and material and frictional properties. Masses and inertias are computed from this information. The dynamics of the system are simulated by integration of the Newton–Euler equations of motion, where each rigid polyhedron is considered to be moving freely under the effect of gravity (and perhaps other body forces) and contact loads [1].

The equations of motion are straightforward and not subject to any explicit constraints, since the constraints are enforced automatically by the contact loads. This is an advantage of the soft contact approach because formulation of the intermittent con-

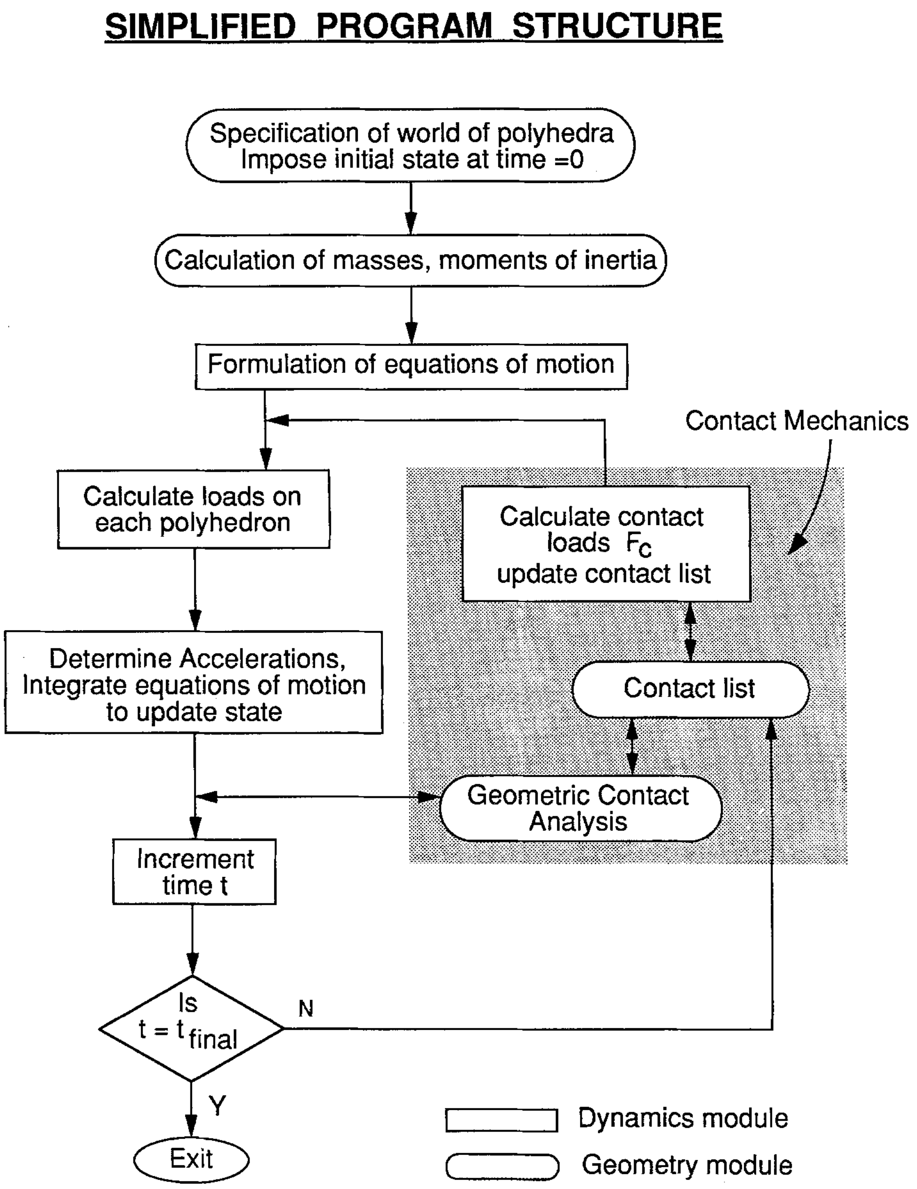

SIMPLIFIED PROGRAM STRUCTURE
Specification of world of polyhedra Impose initial state at time =0
Calculation of masses, moments of inertia
Formulation of equations of motion
Contact Mechanics
Calculate loads on each polyhedron
Calculate contact loads F_C update contact list
Determine Accelerations, Integrate equations of motion to update state
Contact list
Geometric Contact Analysis
Increment time t
Is $t = t_{final}$
N
Y
Exit
Dynamics module
Geometry module

Fig. 1. Simplified overall program structure.

tact constraints for bodies that move while maintaining point contact can be quite complex [10, 11]. However, the contact forces, and thereby the accelerations, can evolve rapidly during contact, leading to stiff differential equations of motion [11]. The loads on every polyhedron being known, their accelerations can be calculated from the equations of motion.

Integration of the accelerations and velocities over a small time interval yields the proposed new state (velocities and positions) of the system at the end of the interval. For stable integration of the stiff differential equations, we use an adaptive time step fourth-order Runge–Kutta scheme. The state of every rigid body is maintained as a list of 13 numbers – 3 for linear velocity, 3 for angular velocity, 3 for position and 4 for orientation in terms of a quaternion. Our integration scheme is a substantially modified version of the algorithm presented in [12]. The proposed new state is passed to the geometry analysis module which determines the corresponding geometric relationship (contacts etc.) between the objects. If there is no change in the topology of contact, i.e. the complete list of VF or EE contacts (see section 3.1) between all pairs of objects, then the proposed state is accepted as a new state and time is incremented. In the case when new contacts are detected, the system interpolates to approximate the time of precise onset of the earliest new contact and the equations of motion are reintegrated up to this time. Broken contacts are dropped from the 'contact list'.

The flow of control during geometric contact analysis is more involved than that shown in the simplified flowchart of Fig. 1. Details of these operations are described below in section 4.

As part of the geometric *contact analysis*, the module produces and updates a *contact list* containing detailed geometric information about all active contacts among the bodies. The dynamics module scans the contact list and calculates contact forces at every contact. The estimation of contact forces from geometric information and material parameters is based on an implementation of the contact model proposed in [1]. The contact loads, in turn, feed into the equations of motion and the integration loop proceeds. In the following section we explain our implemented contact model and the associated material parameters.

3. Dynamics Module

3.1. Control Model

In [1] we presented a conceptual model for formulating material behavior during contact between polyhedral objects. There is no restriction on the size or number of faces in these polyhedra. In principle, objects with smooth surfaces can be approximated by polyhedra with arbitrarily large numbers of faces, so our contact model can apply to all solid objects without restriction. In practice, the number of faces must be constrained to limit the computational effort in the geometry module. It is shown in [1] that at the expense of an arbitrarily small displacement error (either angular or translational), all types of contact between a pair of polyhedral objects can be analyzed as a combination of discrete *vertex–face* (VF) and/or *edge–edge* (EE) contacts. Hence the polyhedral features of primary interest in our discrete model, described below, are vertices, edges and faces.

3.1.1. Vertex–Face Contacts

The microscopic vicinity of each VF contact is modeled through a 'massless mechanism' composed of springs and dampers as shown in Fig. 2. The polyhedron with the vertex has been labeled 1 and the polyhedron with the face as 2. The outside normal to the face is the contact normal $\mathbf{n}$ and the plane of the face defines the tangent plane P_t at contact. Actual contact between the vertex and the face occurs through massless rigid planes called surface elements (SE, see [1]). The SEs are constrained to remain parallel to P_t throughout the contact episode. The contacting surface elements do not penetrate each other and neither do they separate, but they can slip against each other. So all relative motion between the SEs occurs in the plane P_t. Both $\mathbf{n}$ and P_t can be evolving (changing location and/or orientation) in inertial space during a contact episode as the pair of contacting polyhedra move in space.

The SE associated with the vertex is connected to it through two pairs of massless springs and dampers. One of the spring damper pairs, called the normal spring (k_{1_n}) and damper (c_{1_n}), is constrained to be along $\mathbf{n}$. The only deformation permitted for the normal spring is compressive. This is because objects do not pull at each other at contact, but push. The other pair, the tangential spring and damper (k_{1_t} and c_{1_t}), always stays in a plane parallel to P_t. Their orientation in this plane is parallel to the direction of slip between the contacting SEs. A similar arrangement of springs and dampers connects the SE associated with the face. Spring and damper constants for each element can be tuned to model material behavior of the underlying polyhedra, as shown in succeeding sections. Given this mechanism, a typical VF contact would evolve as follows.

At the onset of contact there is no slip (relative tangential velocity) between the SEs, and springs are at their equilibrium lengths. The relative velocity of the vertex with respect to its associated SE causes the intervening springs to deform; similarly, relative velocity of the face with respect to its SE deforms the other set of springs. The component of relative velocity along $\mathbf{n}$ causes the normal springs, k_{1_n} and k_{2_n}, to deform and the tangential component deforms k_{1_t} and k_{2_t}. Deformed springs and velocities at the dampers produce forces that balance the contact force $\mathbf{F}_c$ between the SEs. Since the entire mechanism of springs, dampers and surface elements that exists at a contact is massless, there are no *net forces or moments* acting on it. Net loads on a massless body would yield infinite accelerations. From Coulomb's law of friction, for a given normal force and coefficient of friction the maximum friction force that can be generated between the contacting SEs is the product of the two.

When the force produced by the tangential elements equals the maximum friction force that can be sustained between the SEs, slip occurs. The force produced by the springs and dampers, which equals the contact force between the SEs, is in turn applied to the interacting polyhedra. The point of application C of the contact force is always taken to be the position of the vertex in a VF contact.

The VF contact ceases to exist if either the deformation of both the normal springs becomes tensile or the geometry computations indicate that the vertex has slid off the face.

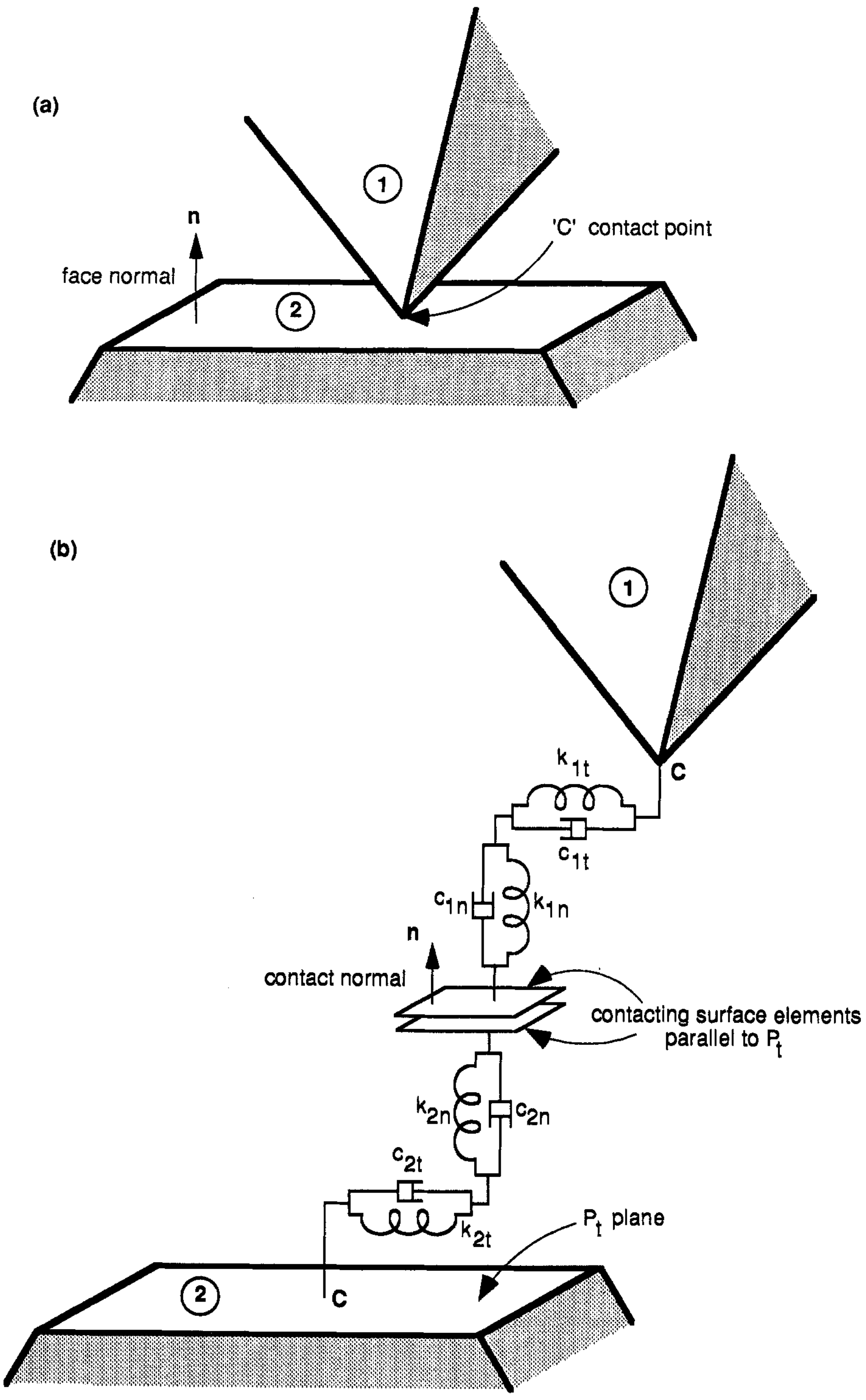


Fig. 2. (a) Schematic drawing of a vertex–face contact. The face and its normal **n** determine the plane of contact and the contact normal.
(b) Magnified view of the microscopic contact device, consisting of springs, dampers and rigid surface elements, at the VF contact.

3.1.2. Edge–Edge Contacts

The microscopic region surrounding an EE contact is modeled identically to that of a VF contact, as shown in Fig. 3. P_t is the plane containing the two edges and **n** is the normal to this plane. The point of application of the contact force is the actual point of contact between the two edges, which is determined by geometric computations and changes as the edges slip against each other.

Each individual VF or EE contact is tracked through four state variables that are appended to the state variable array for the entire system. They are s_{1n}, s_{2n} to track the lengths of the normal springs k_{1n}, k_{2n}, and s_{1t}, s_{2t} to track the lengths of the tangential springs k_{1t}

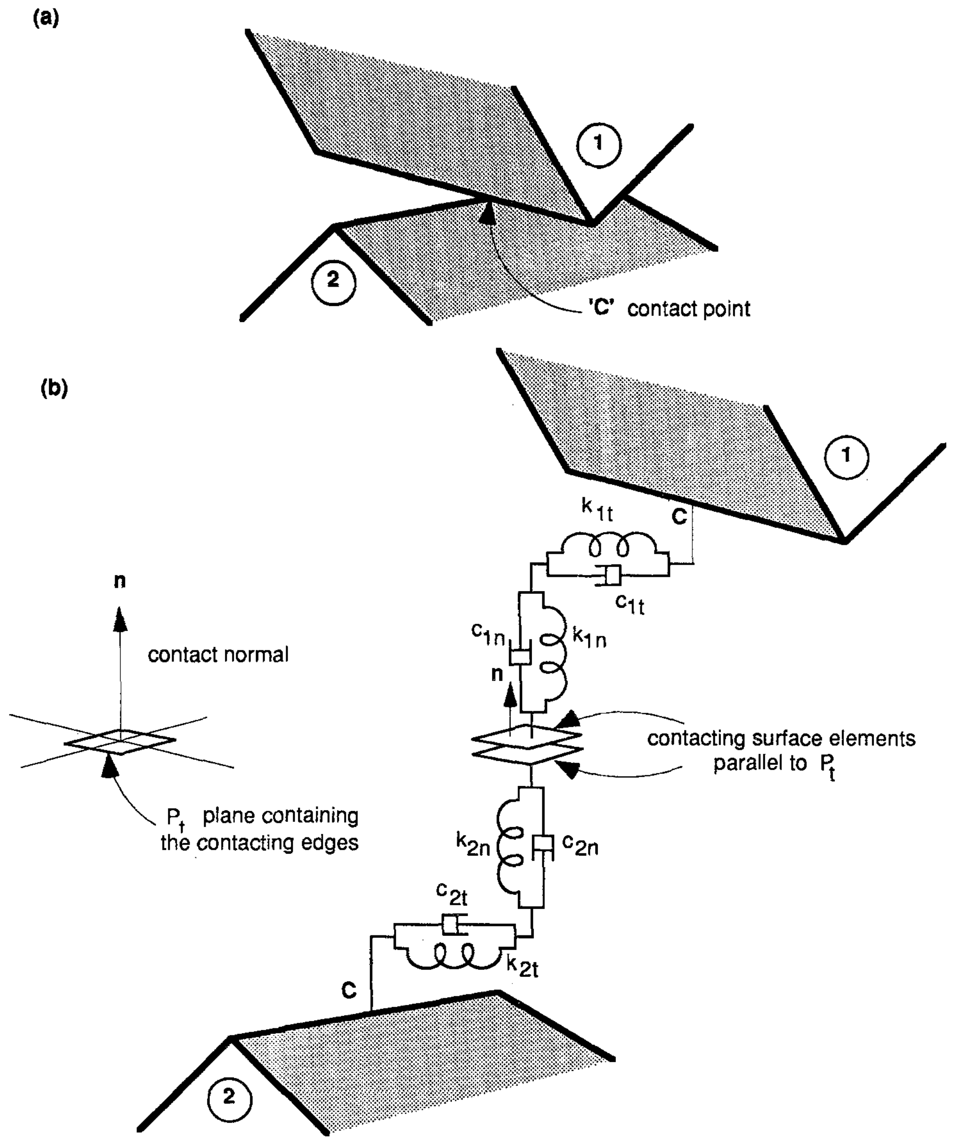


Fig. 3. (a) Schematic drawing of an edge–edge contact. The plane containing the two edges and its normal determine the plane of contact and the contact normal. (b) Magnified view of the microscopic device at the EE contact, analogous to that at a VF contact.

and k_{2t}. They are initialized to zero at beginning of contact. The fact that only compressive deformation of normal springs is allowed implies that $\mathbf{s}_{1n}\cdot\mathbf{n} \le 0$ and $\mathbf{s}_{2n}\cdot\mathbf{n} \ge 0$.

3.1.3. Contact Force and Equations of State

We will now derive expressions for calculating the contact force and the rate of change of the springs in the 'contact mechanism'. The expressions apply with equal validity to both VF and EE contacts. The derivation follows the same steps as in [1]. However, there are some simplifications that arise because of differences between the implemented model and the general model proposed in [1]. The presentation below concentrates mostly on these differences.

The contact force $\mathbf{F}_c$ has two components: the normal component $\mathbf{F}_n$ along $\mathbf{n}$ and the frictional component $\mathbf{F}_t$ in the tangential plane of contact. For reference, we shall follow the schematic of the VF contact of Fig. 2. Here $\mathbf{F}_c$ represents the force that is applied by polyhedron 2 (containing the face) to polyhedron 1 (containing the vertex). It should be noted that, in turn, polyhedron 1 applies a contact force of $-\mathbf{F}_c$ to polyhedron 2. The absolute velocity of the contact point C on polyhedron 1 is denoted by $\mathbf{u}_1$ and on polyhedron 2 by $\mathbf{u}_2$. The absolute velocities of the two SEs are denoted by $\mathbf{w}_1$ and $\mathbf{w}_2$. Figure 4 shows these velocities. Subscripts n and t specify components along $\mathbf{n}$ and in the plane P_t, respectively. Bold letters specify vector quantities, $(\dot{\ }) = \mathrm{d}(\ )/\mathrm{d}t$ and $i = 1, 2$. $\mathbf{F}_n$ and $\mathbf{F}_t$ are given by

$$\mathbf{F}_n = -k_{1n}\mathbf{s}_{1n} - c_{1n}\dot{\mathbf{s}}_{1n} = k_{2n}\mathbf{s}_{2n} + c_{2n}\dot{\mathbf{s}}_{2n} \quad (1)$$

$$\mathbf{F}_t = -k_{1t}\mathbf{s}_{1t} - c_{1t}\dot{\mathbf{s}}_{1t} = k_{2t}\mathbf{s}_{2t} + c_{2t}\dot{\mathbf{s}}_{2t} \quad (2)$$

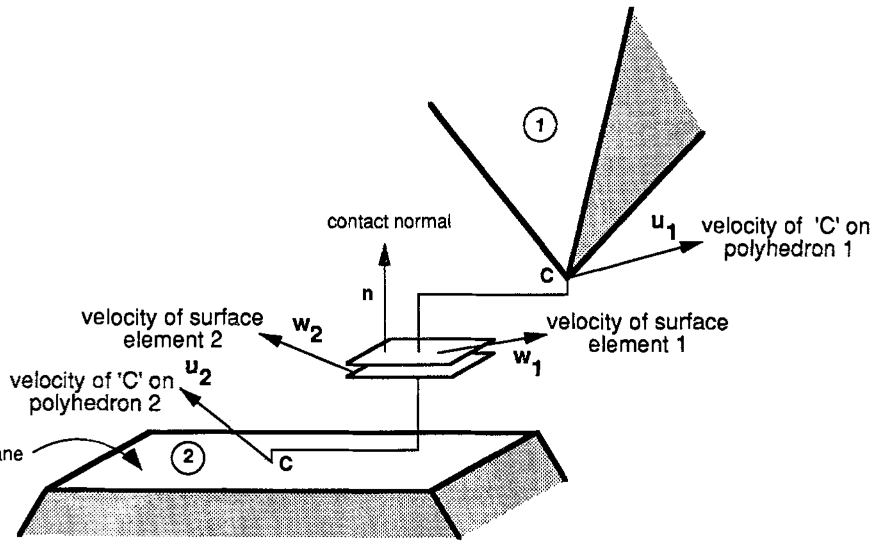

1
contact normal
n
u1 velocity of 'C' on polyhedron 1
C
velocity of surface element 2
w2
velocity of surface element 1
w1
velocity of 'C' on polyhedron 2
u2
Pt plane
2
C

Fig. 4. Schematic drawing of a vertex–face contact showing the various elemental velocities at the contact point C, with $\Delta\mathbf{w}_n = \mathbf{w}_{1n} - \mathbf{w}_{2n} = 0$.

and

$$\dot{\mathbf{s}}_{in} = \mathbf{u}_{in} - \mathbf{w}_{in}, \qquad \dot{\mathbf{s}}_{it} = \mathbf{u}_{it} - \mathbf{w}_{it} \tag{3}$$

With $\Delta\mathbf{u} = \mathbf{u}_1 - \mathbf{u}_2$, $\Delta\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_2$ and $\Delta\mathbf{w}_n = 0$ (no relative motion between the SEs along $\mathbf{n}$), substitution into equations (3) yields

$$\dot{\mathbf{s}}_{1n} - \dot{\mathbf{s}}_{2n} = \Delta\mathbf{u}_n, \qquad \dot{\mathbf{s}}_{1t} - \dot{\mathbf{s}}_{2t} = \Delta\mathbf{u}_t - \Delta\mathbf{w}_t \tag{4}$$

To proceed further, we have to branch into the various cases delineated below.

3.1.4. *All Dampers Non-zero*

This is the case when $c_{1n}, c_{2n}, c_{1t}, c_{2t} > 0$. Through manipulation of equations (1), (3) and (4) we obtain

$$\mathbf{F}_n = -\mathbf{b}_n, \qquad \mathbf{F}_t = -\mathbf{b}_t + c^*\Delta\mathbf{w}_t \tag{5}$$

where

$$\left.\begin{aligned} c^* &= \frac{c_{1t}c_{2t}}{c_{1t}+c_{2t}},\\ \mathbf{b}_n &= \frac{1}{c_{1n}+c_{2n}}(c_{2n}k_{1n}\mathbf{s}_{1n} - c_{1n}k_{2n}\mathbf{s}_{2n} + c_{1n}c_{2n}\Delta\mathbf{u}_n)\\ \mathbf{b}_t &= \frac{1}{c_{1t}+c_{2t}}(c_{2t}k_{1t}\mathbf{s}_{1t} - c_{1t}k_{2t}\mathbf{s}_{2t} + c_{1t}c_2\Delta\mathbf{u}_t) \end{aligned}\right\} \tag{6}$$

If μ is the coefficient of friction between the two surfaces and λ is a non-negative real variable, then the friction law can be expressed as:

$$|\mathbf{F}_t| \le \mu|\mathbf{F}_n|, \qquad \Delta\mathbf{w}_t = -\lambda\mathbf{F}_t, \qquad \lambda \ge 0 \tag{7}$$

In equations (5), (6) and (7), c^* is a constant, and $\mathbf{b}_n$ and $\mathbf{b}_t$ can be calculated from known information at every time step. This implies that $\mathbf{F}_n$ is known. The only remaining unknowns are $\mathbf{F}_t$, $\Delta\mathbf{w}_t$ and λ. Substitution of equations (5) and (6) into equations (7) gives:

$$\frac{|\mathbf{b}_t|}{(1+\lambda c^*)} \le \mu|\mathbf{b}_n| \tag{8}$$

As in [1], at every time step we first check if $|\mathbf{b}_t| \le \mu|\mathbf{b}_n|$. If so, then the SEs are in a state of stick and we set $\lambda = 0$ and $\Delta\mathbf{w}_t = 0$. This implies that $\mathbf{F}_t = -\mathbf{b}_t$ can be calculated.

If $|\mathbf{b}_t| > \mu|\mathbf{b}_n|$, then we determine the value of λ that makes $|\mathbf{b}_t|/(1+\lambda c^*) = \mu|\mathbf{b}_n|$, which is:

$$\lambda = \frac{|\mathbf{b}_t| - \mu|\mathbf{b}_n|}{c^*\mu|\mathbf{b}_n|} \tag{9}$$

Once λ is known, the other unknowns can be calculated from

$$\mathbf{F}_t = -\frac{\mathbf{b}_t}{(1+\lambda c^*)}, \qquad \Delta\mathbf{w}_t = \frac{\lambda\mathbf{b}_t}{(1+\lambda c^*)} \tag{10}$$

The state of the springs in the contact mechanism is tracked with the following differential equations, obtained from equations (1), (2) and (4):

$$\dot{\mathbf{s}}_{1n} = \frac{c_{2n}}{c_{1n}+c_{2n}}\Delta\mathbf{u}_n - \frac{1}{c_{1n}+c_{2n}}(k_{1n}\mathbf{s}_{1n} + k_{2n}\mathbf{s}_{2n}),$$

$$\dot{\mathbf{s}}_{1t} = \frac{c_{2t}}{c_{1t}+c_{2t}}(\Delta\mathbf{u}_t - \Delta\mathbf{w}_t) - \frac{1}{c_{1t}+c_{2t}}(k_{1t}\mathbf{s}_{1t} + k_{2t}\mathbf{s}_{2t})$$

$$\dot{\mathbf{s}}_{2n} = \frac{-c_{1n}}{c_{1n}+c_{2n}}\Delta\mathbf{u}_n - \frac{1}{c_{1n}+c_{2n}}(k_{1n}\mathbf{s}_{1n} + k_{2n}\mathbf{s}_{2n}),$$

$$\dot{\mathbf{s}}_{2t} = \frac{-c_{1t}}{c_{1t}+c_{2t}}(\Delta\mathbf{u}_t - \Delta\mathbf{w}_t) - \frac{1}{c_{1t}+c_{2t}}(k_{1t}\mathbf{s}_{1t} + k_{2t}\mathbf{s}_{2t})$$

3.1.5. Zero Damping

In the extreme case of completely elastic contacting materials (no damping), $\Delta \mathbf{w}_t$ cannot be determined uniquely without adding extra equations and is hence set to zero. The contact model becomes:

$$\mathbf{F}_n = -k_{1n}\mathbf{s}_{1n} = k_{2n}\mathbf{s}_{2n}, \qquad \dot{\mathbf{s}}_{1n} = \frac{k_{2n}}{k_{1n} + k_{2n}}\Delta\mathbf{u}_n,$$

$$\dot{\mathbf{s}}_{2n} = \frac{-k_{1n}}{k_{1n} + k_{2n}}\Delta\mathbf{u}_n, \qquad \mathbf{F}_t = -k_{1t}\mathbf{s}_{1t} = k_{2t}\mathbf{s}_{2t}$$

subject to

$$|\mathbf{F}_t| \le \mu|\mathbf{F}_n|, \qquad \dot{\mathbf{s}}_{1t} = \frac{k_{2t}}{k_{1t} + k_{2t}}\Delta\mathbf{u}_t,$$

$$\dot{\mathbf{s}}_{2t} = -\frac{k_{1t}}{k_{1t} + k_{2t}}\Delta\mathbf{u}_t \quad \text{and} \quad |\mathbf{s}_{it}| \le \frac{\mu|\mathbf{F}_n|}{k_{it}}$$

3.1.6. Both Non-zero and Zero Dampers

For contacts that have a mix of non-zero and zero dampers, the equations are combinations of the two cases cited above. For example, for the case of $c_{1n} = c_{2t} = 0$ and $c_{2n}, c_{2t} \neq 0$, the model is:

$$\mathbf{F}_n = -\mathbf{b}_n = -k_{1n}\mathbf{s}_{1n},$$

$$\dot{\mathbf{s}}_{1n} = \Delta\mathbf{u}_n - \frac{1}{c_{2n}}(k_{1n}\mathbf{s}_{1n} + k_{2n}\mathbf{s}_{2n}),$$

$$\dot{\mathbf{s}}_{2n} = -\frac{1}{c_{2n}}(k_{1n}\mathbf{s}_{1n} + k_{2n}\mathbf{s}_{2n})$$

$$\mathbf{F}_t = -k_{1t}\mathbf{s}_{1t} = k_{2t}\mathbf{s}_{2t} + c_{2t}\dot{\mathbf{s}}_{2t}$$

subject to

$$|\mathbf{F}_t| \le \mu\mathbf{F}_n|, \qquad \dot{\mathbf{s}}_{1t} = \Delta\mathbf{u}_t - \frac{1}{c_{2t}}(k_{1t}\mathbf{s}_{1t} + k_{2t}\mathbf{s}_{2t}),$$

$$\dot{\mathbf{s}}_{2t} = -\frac{1}{c_{2t}}(k_{1t}\mathbf{s}_{1t} + k_{2t}\mathbf{s}_{2t})$$

Now we tackle the issue of tuning our model parameters, the spring and damper constants, to reflect material behavior.

3.2. Specifying Material Parameters

The constituting elements of our model simulate specific object responses during contact. Their values can also be set to 'match' certain material properties of the interacting objects. Springs (elastic elements) keep the bodies from inter-penetrating, and accomplish momentum transfer through absorption and release of kinetic energy. They also model the 'elastic feel', i.e. the hardness or the softness of the material, and can be compared to the Young's modulus for the material [13]. Energy dissipation is accomplished through dampers and friction. The damper constants can be related to the coefficient of restitution or the energy loss that occurs during an impact.

3.2.1. Spring Constants

All objects in our simulation are modeled as 'hard' rigid objects. Hence spring stiffnesses are set quite high. In particular, normal spring stiffnesses k_n are based on permissible deformation and the required energy absorption capacity. If E_{max} is the energy to be stored for a spring deflection of δ_{max}, then energy balance gives:

$$k_n = \frac{2E_{max}}{\delta^2_{max}} \tag{11}$$

E_{max} is the maximum kinetic energy that the body can have and is determined by its mass (m) and maximum velocity (v_{max}) as $E_{max} = \frac{1}{2}mv^2_{max}$. The allowable spring deflection δ_{max} is set to be a small fraction of some representative linear dimension (average dimension) of the object. Small spring deflections are desirable for geometric reasons: they lead to robust geometric contact analysis algorithms and a realistic visual perception of interacting rigid bodies in the displayed animations.

In the interests of isotropy, k_t is set equal to k_n although it can just as easily be set to a different value. Setting damper constants is a bit more involved.

3.2.2. Damper Constants

Even in the absence of friction, energy loss during impact is difficult to quantify simply. It is a complex function of material properties, geometry of contact, duration of impact, etc. Traditionally, a non-zero coefficient of restitution e_r is used to account for energy lost during impact ([14]). For setting the normal damper constant c_n to achieve a desired energy loss for a material, one can consider an experiment involving a frictionless impact between a sphere of mass m falling on a stationary horizontal plate made of the same material. Since the sphere and plate have identical material properties, their interaction is equivalent to the single spring–mass–damper system of Fig. 5. The single spring and damper shown in Fig. 5 have *half the stiffness and damping* of the actual material. If v_a is the velocity of the sphere after impact and v_b before impact, then the energy retained during impact, E_r, and the coefficient of restitution, e_r, are related by:

$$E_r = \frac{\frac{1}{2}mv^2_a}{\frac{1}{2}mv^2_b} = \frac{v^2_a}{v^2_b} = e^2_r \tag{12}$$

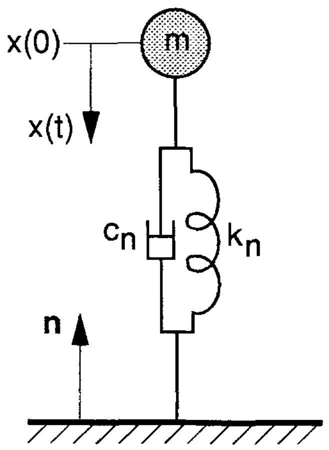


Fig. 5. Single spring–mass–damper system.

Referring to Fig. 5, if $x(t)$ is the displacement of the sphere from the equilibrium position of the normal contact spring, then it feels a force $F = -c_n\dot{x} - k_n x$. Its equation of motion is:

$$m\ddot{x} + c_n\dot{x} + k_n x = 0 \tag{13}$$

With natural frequency

$$\omega_n = \sqrt{\frac{k_n}{m}},$$

damping ratio

$$\zeta = \frac{c_n}{2m\omega_n}, \quad (0 \le \zeta \le 1),$$

and

$$\omega_d = \omega_n\sqrt{1-\zeta^2},$$

the solution to (13) is

$$x(t) = \frac{v_b}{\omega_d} e^{-\zeta\omega_n t}\sin(\omega_d t).$$

Contact between the sphere and plate is broken when $F(t) = 0$. This occurs at time t^* such that

$$\left.\begin{aligned} \tan(\omega_d t^*) &= -\frac{2\zeta\sqrt{1-\zeta^2}}{1-2\zeta^2}, \\ E_r = e^{-2\zeta\omega_n t^*} &\Rightarrow t^* = -\frac{\ln(E_r)}{2\zeta\omega_n} \end{aligned}\right\} \tag{14}$$

Manipulation of equation (14) gives

$$\ln(E_r) = -\frac{4\zeta}{\sqrt{1-\zeta^2}}\tan^{-1}\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right) \tag{15}$$

The transcendental equation (15) can be solved, numerically, for c_n. Equation (15) is plotted in Fig. 6. It shows that $\zeta = c_n = 0$ gives a completely elastic impact, i.e. $E_r = 1$ and there is no dissipation of energy. However, $\zeta = 1.0$ does not yield the expected plastic impact where the bodies do not separate after impact (i.e. $v_a = 0$). This is because we break contact at $F(t) = 0$; with non-zero k_n and c_n, this always occurs with $v_a > 0$. In fact, equation (15) shows that, $\lim_{\zeta\to 1}\ln(E_r) = -4$, or $E_r \approx 0.01832$. A truly plastic impact, if desired, can be obtained by setting $k_n = 0$ and with high damping.

The tangential damper c_t, along with friction, damps energy in the P_t plane. In our implementation, its primary function is to damp out frictional oscillations. Hence it is either set from the energy analysis of equation (15) or to the critical damping value of $\sqrt{4mk_t}$ for shortest time to attain equilibrium in the P_t plane.

It is important to note, however, that the preceding energy loss analysis is limited to the idealized impact shown in Fig. 5. In a generalized 3-D impact, actual energy loss in the simulation is a complicated function of several factors: the normal and tangential damper constants of both the contacting bodies, their *effective masses* at the point of contact and the coefficient of friction. As shown in Appendix A, effective mass is a function of geometry and inertia distribution and can change even during the contact episode as a result of relative rotation of the contacting bodies. Because of all these complex factors, a simple expression for energy loss at impact in terms of spring and damper constants is usually not possible.

3.3. Estimating and Controlling the Time Response of Contact Behavior

Spring–mass–damper systems, like that of Fig. 5, have natural frequencies associated with their response [15]. Meaningful numerical integration requires that the time step of integration Δt be several times smaller than the smallest time period $\tau_{\min}$ (or time constant) associated with these natural frequencies.

Among other things, these time constants depend on the ratio of effective masses to material stiffnesses for each body. The minimum effective mass $m_{\min}$ for a body, depending upon its geometry as shown in Appendices A and B, can become extremely small. Material stiffnesses are usually quite high because we model the bodies as hard rigid objects. These two factors can lead to an arbitrarily small $\tau_{\min}$, requiring an even smaller Δt for stable integration. Very small time steps are undesirable for two reasons. Firstly,

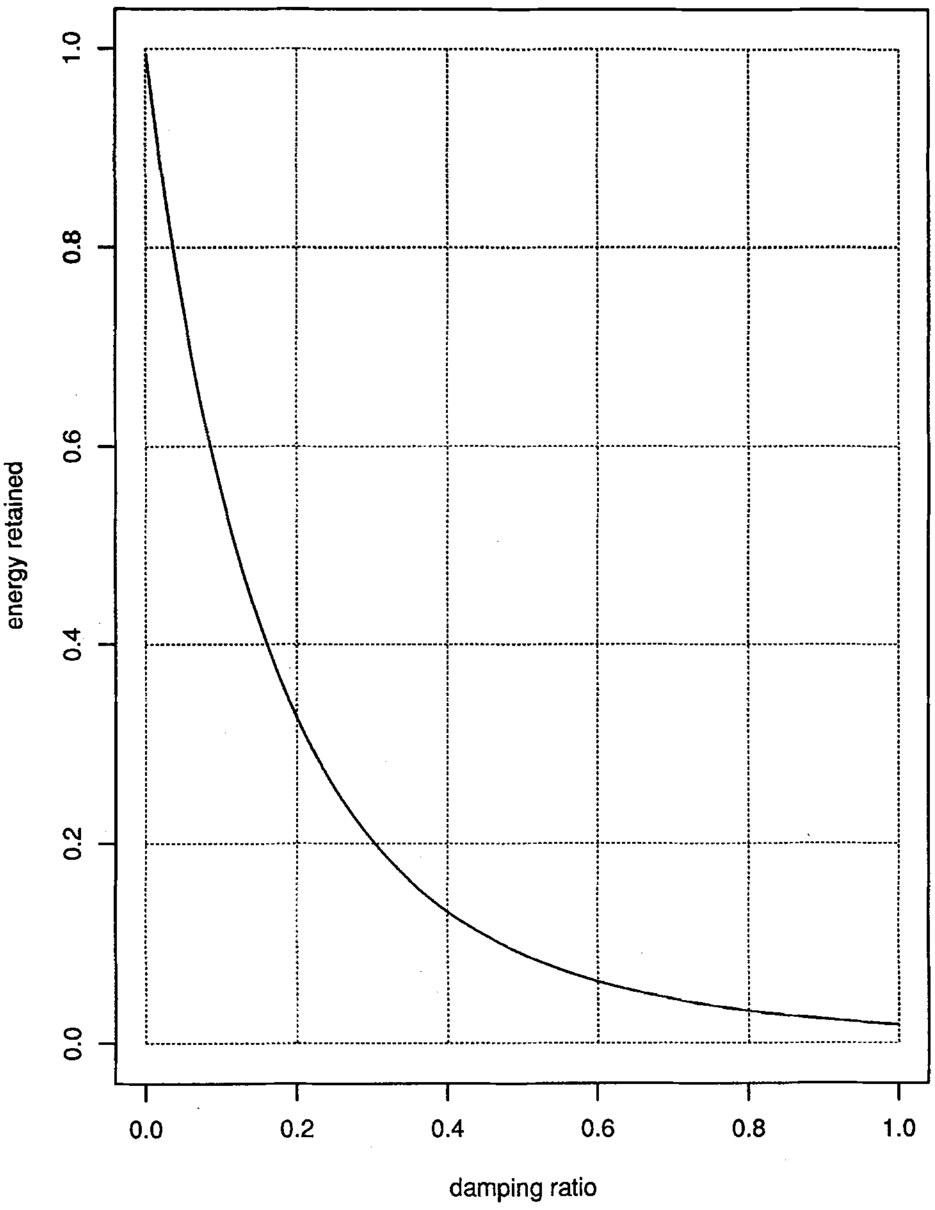


Fig. 6. Graph of energy retained E_r versus damping ratio ζ for tuning the normal damper c_n.

program solution speed is affected adversely because smaller time steps require more steps in the simulation and hence more computation. Secondly, very small time steps can lead to a destabilizing accrual in round-off errors in numerical computations.

To address these problems we limit the minimum allowed time step ($\Delta t_{\min}$) and suppress behavior that cannot be integrated meaningfully with it. Non-integrable high-frequency model response is suppressed by altering appropriate model parameters. Broadly, there are two types of time constants that we check for: those that are associated independently with the material for each body and time constants that arise because of interaction between materials.

3.3.1. Time Constants associated with Each Body

Based on the material model that we implement for contact, each body has a set of two spring–mass–damper systems – one in the P_t plane and the other along **n**. From the solution of equation (13) it can be seen that the response of the spring–mass–damper system is governed by two time constants, $1/\zeta\omega_n$ and $2\pi/\omega_n$. In terms of material stiffness and minimum effective masses $m_{\min}$ (calculated as shown in Appendices A and B), they are $\tau = 2\pi\sqrt{m_{\min}/k}$ and $\tau = 2(m_{\min}/c)$. The program checks to see that $\Delta t_{\min}$ is less than these values of τ for $k = k_n$, $k = k_t$, $c = c_n$ and $c = c_t$. If $\Delta t_{\min}$ is greater in any of these cases, the corresponding stiffness/damping is decreased appropriately.

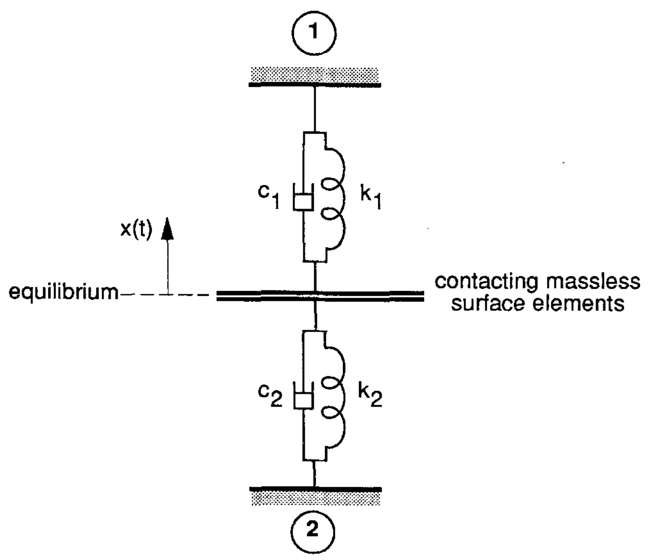


Fig. 7. Analysis of response of contacting surface elements.

However, the actual time constants that the integrator experiences are the pairwise material interaction time constants at contact, as described below.

3.3.2. Time Constants at Contact

The contact models of Figs 2 and 3, either along **n** or in the P_t plane, can be analyzed as an interacting system of two pairs of spring–mass–dampers (see Fig. 7). The equation of motion for such a system can be obtained as a third-order linear differential equation. When contact is made the characteristic equation associated with this differential equation can be solved numerically to estimate its time constants. However, if the stiffnesses and masses of the contacting bodies are comparable, then most of the time constants for the contact model are close to the time constants estimated for the individual bodies, as obtained above.

One of the time constants also involves the response of the massless contacting surface elements when subjected to the spring and damper forces from both bodies, and it can be obtained as the response of the elements shown in Fig. 7. If $x(t)$ is the displacement of the contacting surface elements from their equilibrium position then their equation of motion is given by

$$(c_1 + c_2)\dot{x} + (k_1 + k_2)x = 0 \qquad (16)$$

It should be noted that since the contacting surface elements are massless they should always be at their equilibrium position, but numerical discretization can lead to drift from this position. The solution to equation (16) is an exponential decay which has the form

$$x(t) = x(0)\,\mathrm{e}^{-t/\tau}, \qquad \tau = \frac{c_1 + c_2}{k_1 + k_2} \qquad (17)$$

Our software system checks to make sure that $\Delta t_{\min}$ is less than τ, both along **n** and in the P_t plane. If $\Delta t_{\min} > \tau$ in the normal direction, we set the normal dampers to be zero ($c_{1_n} = c_{2_n} = 0$). If the same holds true in the P_t plane, we decrease the tangential stiffnesses (k_{1_t} and k_{2_t}) of the contact model appropriately. Incidentally, these corrections can lead to a substantial difference in the outcome of impacts involving marginally damped material (from the expected outcome in terms of energy loss) because the low damping specified for the materials in the normal direction jumps to zero damping, implying elastic material. And low damping specified in the tangential direction can jump to much higher effective damping when the corresponding stiffnesses are reduced. The above analysis does not apply, and is not necessary, for the damperless case. Besides estimating and controlling the time response at contact, our integration scheme performs some additional adjustments to achieve and enhance stability, as described below.

3.4. Additional Adjustments for Numerical Stability

Finite precision dictates that non-zero thresholds be set for deciding different cases, for instance choosing between an elastic and a damped impact for very small damper values. Contact springs are 'adjusted' at the end of every time step. The normal and tangential springs are realigned to lie along the updated **n** and P_t plane respectively. In addition, the direction of the tangential springs is made parallel to the direction of slip between the bodies and their magnitude kept within the *friction cone.*

Throughout the integration, normal contact springs are never allowed to have tensile deformation. In addition, the integration time step is refined adaptively for several causes: reversal in spring deformation directions, 'zero crossings' of tangential springs, velocity reversals, etc.

4. Geometry Module

4.1. Object Representation

The major geometric computation addresses the question of exactly where bodies make contact. Here a representation decision must be made, since determining spatial relationships depends on exactly how the objects are represented geometrically. For example, algebraic surfaces could be used to represent the objects. Contacts would then be determined by solving pairs of non-linear algebraic equations, possibly of high order. Analytic solutions of such equations are usually not possible, and iterative techniques must be used that can be quite slow computationally.

The representation we have chosen is that of convex polyhedra. This restricts the complexity of objects that can be simulated for two reasons. Firstly, some objects one might want to include in a simulation are not convex. Secondly, smoothly continuous surfaces, such as spheres and cylinders, cannot be represented exactly. Smooth surfaces can be approximated closely (*closely* in the sense of spatial error between a point in the polyhedron and the corresponding point of the modeled smooth object – discontinuities in surface normals that occur at edges can never be eliminated, although their magnitude can be limited) if desired, by using large numbers of faces in the polyhedron. Issues of computational performance limit the number of faces that can be used in a practical application. However, in favor of polyhedral representations is the fact that most geometric calculations only require solutions of linear equations. Since similar computations are required for all faces (or all edges or all vertices) of an object, many similar simple computations are performed. In principle, this leads to a form of problem that is well suited for solution by massively parallel computers. Advances in computer architecture and VLSI processing speeds that will occur during the next few years can be expected to result in real-time simulation capabilities for many problems of interest.

Before contact analysis begins, the geometry module computes the invariant inertial properties (masses and moments of inertia) of the convex polyhedra in the simulation, based on their geometries. During the simulation, a detailed description of the body's shape and location (the *location* of an object specifies all six of its degrees of freedom – 3 Cartesian coordinates for its center of mass and 3 orientation angles) is used to determine when and where contacts occur.

4.2. Integration/Contact-analysis Loop

Integrating the equations of motion is an initial value problem. (Figure 8 explains the computational flow of contact analysis as it supplements this integration.) From the known *system-state* at time t_0, we integrate forward to time $t_1 = t_0 + \mathrm{d}t$. If the integration step is valid, we now have the system-state at t_1, and the process can be repeated. The system state at time t_0 has two components: the *dynamic-state*, D_0, and the *contact-state*, C_0. D_0 specifies the positions and velocities (both linear and rotational) of all objects at t_0. C_0 specifies the state of each contact that exists at t_0 including, for each contact, which objects are involved and the exact state of the contact (e.g. the edges and vertices involved, and the location and status of the contact). The contact-state is represented by a list of *contact-blocks*, each specifying the above information for one contact. Based on C_0, the dynamic equations, E_0, at time t_0 can be specified including all active inter-object spring and damper forces. We now integrate forward, using E_0, to time t_1. The result at t_1 will be a new dynamic-state, D_1, and a corresponding new contact-state, C_1. Contact analysis at t_1 determines C_1, which can then be compared with C_0. If the set of contacts in C_1 and C_0 are the same, then no change in contact-state occurred, so C_1 and D_1 can be accepted as valid starting points for the next integration step (the inner loop shown in Fig. 8).

On the other hand, if C_1 and C_0 differ in the set of contacts they contain, then either a *new* contact appeared or an existing contact disappeared (i.e. was *broken*) during the integration interval. *Broken* contacts are normally determined by the dynamics module based on when the normal force acting at the contact goes to zero. (For certain exception conditions described later, geometric conditions must be used to determine *broken* contacts.) Such an event is easily handled by removing the corresponding contact-block from the contact-state list. *New* contacts, however, have to be analyzed in detail. In particular, the earliest *new* contact event during the dt interval is determined. Assuming this contact occurs at some fraction δ of the whole interval ($0 \leq \delta \leq 1$), we can integrate forward to time $t_c = t_0 + \delta \cdot \mathrm{d}t$. The resulting dynamic-state, D_c, and contact-state, C_c, at t_c can be accepted as valid starting points for the next integration step, since no contact event occurred during the interval $t_0 \leq t \leq t_c$ (the outer loop shown in Fig. 8).

All contacts between the polyhedra must be identified and tracked. At every time step, the geometry module accomplishes this through *contact analysis*. Contact analysis consists of *interference detection* (detecting whether objects are in contact) and maintaining the interference information in a list of active point contacts (the *contact-state*) that the dynamics module uses to calculate contact forces.

4.3. Interference Detection

The interference detection algorithm [16] looks for a *separating plane* that partitions space with all vertices of one object (say, object **A**) on one side of the separating plane and all vertices of the other object

D_0	Dynamic state at time t_0, the start of an integration step
C_0	Contact state at time t_0 based on D_0
E_0	Dynamics equations (determined by C_0)
D_1	Proposed dynamic state after integrating forward to time $t_1 = t_0 + dt$
C_1	Proposed contact state at time t_1 based on D_1
δ	Fraction of step size dt at which first new contact occurs (-1 if no new contacts)
D_c	Dynamic state after integrating forward to time of first new contact, $t_c = t_0 + \delta \cdot dt$
C_c	Contact state at time t_c based on D_c

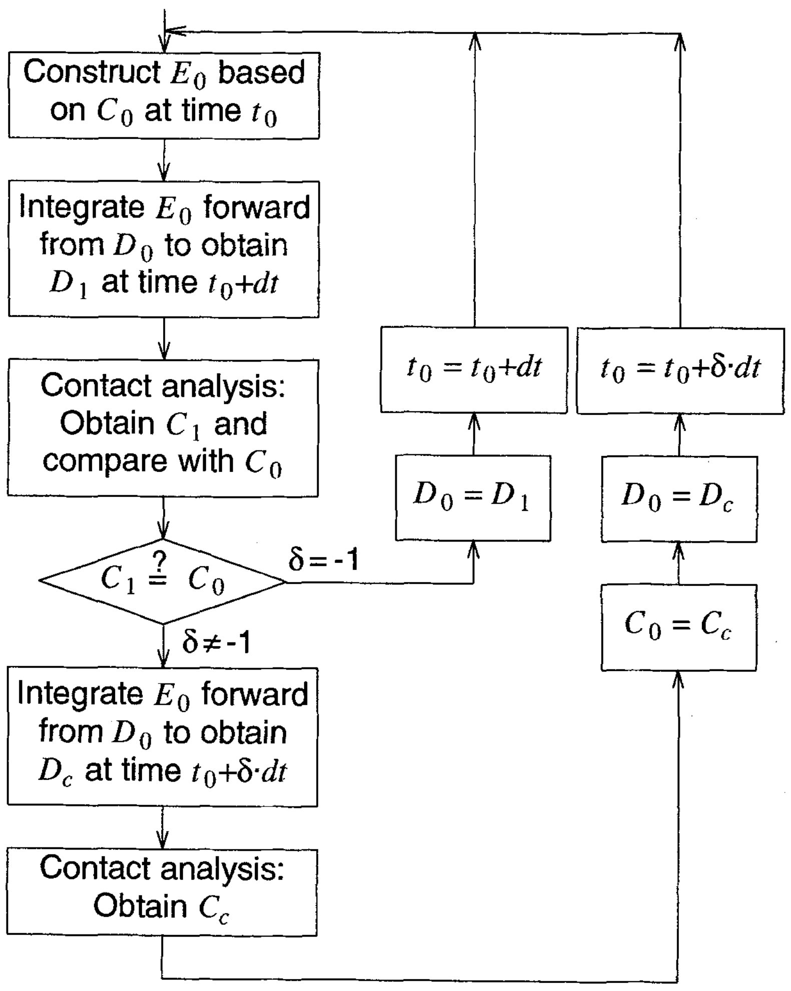

Construct E_0 based on C_0 at time t_0
Integrate E_0 forward from D_0 to obtain D_1 at time t_0+dt
Contact analysis: Obtain C_1 and compare with C_0
$C_1 \overset{?}{=} C_0$
$\delta = -1$
$\delta \neq -1$
Integrate E_0 forward from D_0 to obtain D_c at time $t_0+\delta \cdot dt$
Contact analysis: Obtain C_c
$t_0 = t_0+dt$
$D_0 = D_1$
$t_0 = t_0+\delta \cdot dt$
$D_0 = D_c$
$C_0 = C_c$

Fig. 8. Simplified flow diagram of computation.

(say, object **B**) on the other side of the plane. Each edge of **A** is examined in turn to see if it supports such a separating plane, i.e. if there is a plane that contains the edge and that is a separating plane. As soon as such a plane is found, the algorithm terminates and reports success. If no separating plane is supported by any edge of **A**, the process is repeated, but this time looking for a separating plane supported by an edge of **B**. Again, success is reported as soon as such an edge is found. If no such edge is found, the algorithm reports failure, i.e. that the objects interfere with each other.

Since each edge is considered in turn, and for each edge all vertices of the other object are considered, the algorithm is of complexity $\mathbf{O}(n^2)$ in the number of vertices of the objects involved. (For convex polyhedra, the number of object edges E, faces F and vertices V are related linearly by $E = F + V - 2$, so n can be used generically to represent the number of edges, faces or vertices in complexity expressions.) In principle, each possible pair of objects in the scene must be examined. Again, this requires $\mathbf{O}(k^2)$ comparisons, where k is the number of objects in the scene. The overall complexity is the product of these two complexity results, or $\mathbf{O}(N^2)$ comparisons, where N is the total number of vertices in all objects being considered.

Fortunately, it is possible to reduce the amount of computation required for interference detection substantially. Firstly, objects are divided into two classes: *moving* and *fixed*. Fixed objects are used to represent stationary obstacles in the simulation space or walls bounding the possible motion of objects. Interference computations are not needed between fixed objects. Only *moving–moving* and *moving–fixed* object comparisons are made. Secondly, very fast *bounding-box* screening tests are performed on candidate object pairs before detailed interference calculations are undertaken. These tests look for separating planes that are orthogonal to the world coordinate reference frame axes. Since bounding-box extents are stored in the data structure representing an object's geometry, these tests are done in constant time for each object pair, independently of the number of vertices. The bounding-box extents must be updated for every moving object at each time step, an amount of computation that grows only linearly with the number of objects. The number of bounding-box tests still grows as the square of the number of moving objects. The relatively expensive separating plane test is only performed on the smaller number of object pairs not eliminated by the bounding-box test.

4.4. Determining the Contact-state

Interference detection (often called collision detection) is computationally the most expensive part of contact analysis, but much of the useful information for the force model is obtained by determining the contact-state. Interesting new problems arise in the context of generating and updating the contact-state.

Many pairs of objects may be in contact simultaneously, and an object may be involved in several contacts simultaneously, either with one or several other objects. For example, an FF contact between two cubes will appear as four VF contacts if all the contacting face vertices of the smaller cube lie within the contacted face of the larger cube. Or, such a contact may appear as two VF contacts and two EE contacts if the contacting face of the smaller cube extends past an edge of the face of the larger cube. Each such VF or EE contact is identified by its *contact-id*, which is a unique 6-tuple of information:

1. The index of the first object, say object **A**, among all possible objects.
2. The index of the second object, say object **B**, among all possible objects.
3. The type of the contacting element of **A** – F (face), V (vertex) or E (edge).
4. The type of the contacting element of **B** – F (face), V (vertex) or E (edge).
5. The index of the contacting element among all elements of that type in **A**.
6. The index of the contacting element among all elements of that type in **B**.

Information describing each contact is stored in a data structure called a *contact-block*. The contact-block contains the contact-id and additional information about the contact, including:

1. The location of the point of contact.
2. The direction along which the normal impact force acts. For VF contacts, this is the direction of the normal to the contacted face. For EE contacts, this direction is obtained by taking the cross product of the two edge-vectors involved.
3. The type of the contact: *new* if initial contact occurred after the beginning of the previous integration step; or *continuing* if the contact existed at the beginning of the previous integration step.

The *contact-state* of the system consists of a list of contact-blocks, one for each active contact in the system, ordered by contact-id. To maintain the contact-state list, it is necessary to determine when a contact first occurs (*initial-contact*) and when it ends (*broken-contact*), i.e. when the two objects separate at that specific contact. These occurrences are known as *contact-events*. Initial-contacts are determined by purely geometric considerations. When an initial-contact occurs, a new contact-block is added to the contact-state list. Broken-contacts are determined and deleted from the contact-state list based on either dynamic or geometric considerations, as explained later.

4.4.1. Determining Initial Contacts

When two objects approach each other and their motion is approximated by integration with a finite time step, actual contact will, in general, occur sometime during the integration interval. Since the behavior of the contacting objects can be extremely sensitive to exactly when and where each contact occurs, it is essential to approximate the time and position of *initial contact* accurately. This could be done by an iterative process such as time-halving the integration steps. Unfortunately, such trial-and-error approaches can require many iterations to reach an acceptably accurate result.

To reduce the amount of computation, we replace iteration by interpolation. Consider the case of a VF contact where vertex $\mathbf{v}$ of object **A** strikes the face of object **B** whose normal points in the direction $\mathbf{n}$. At the start of the integration step, time t_1, the objects do not interfere. $\mathbf{v}$ is at location $\mathbf{v}_1$ and face **F** is at location $\mathbf{F}_1$ with face normal $\mathbf{n}_1$. At time t_2, the end of the proposed integration interval, $\mathbf{v}$ has penetrated **F**, and the corresponding quantities are $\mathbf{v}_2$, $\mathbf{F}_2$ and $\mathbf{n}_2$. At some intermediate time t_c, the objects must have experienced initial contact, with $\mathbf{v}$, **F** and $\mathbf{n}$ assuming values of $\mathbf{v}_c$, $\mathbf{F}_c$ and $\mathbf{n}_c$. Contact analysis then determines t_c through linear interpolation. The method used is described in Appendix C.

4.4.2. *Tracking Contacts*

Contacts must be tracked, i.e. their positions and orientations must be determined, through several (possibly many) integration steps for two reasons. Firstly, even during impacts where bodies rapidly bounce off each other, the contact model requires several time steps to stably integrate the rapidly evolving forces. Secondly, contacts of extended duration will occur when bodies slide along each other. The integration time step is automatically adjusted to a small value upon encountering a new contact. The dynamics integrator then gradually increases the time step within the limits of accuracy constraints.

VF and EE contacts are tracked differently. VF contacts are tracked by the dynamics module. The *vertex* position and *face normal* vector involved in a VF contact are reported to the dynamics module in a *new* contact-block when the contact is first detected. At t_0, the time at which the new contact occurred, the following information is known:

$\mathbf{R}_{v0}$	rotation specifying the orientation of the object whose vertex is involved in the contact (from the dynamic-state at t_0);
$\mathbf{r}_0$	location of the center of mass of the above object (from the dynamic-state at t_0);
$\mathbf{v}_0$	location of the contacting vertex (from the contact-block at t_0);
$\mathbf{R}_{f0}$	rotation specifying the orientation of the object whose face is involved in the contact (from the dynamic-state at t_0);
$\mathbf{n}_0$	normal to the contacted face (from the contact-block at t_0).

The dynamics module can independently determine the contacting vertex position and face normal orientation at any subsequent time t_1. Using object rotations $\mathbf{R}_{v1}$ and $\mathbf{R}_{f1}$ and center of mass $\mathbf{r}_1$, which are known from the dynamic-state at t_1, the vertex position $\mathbf{v}_1$ and face normal $\mathbf{n}_1$ at $\mathbf{t}_1$ are:

$$\mathbf{v}_1 = \mathbf{r}_1 + \mathbf{R}_{v1}\mathbf{R}_{v0}^{-1}(\mathbf{v}_0 - \mathbf{r}_0) \qquad (18a)$$

$$\mathbf{n}_1 = \mathbf{R}_{f1}\mathbf{R}_{f0}^{-1}\mathbf{n}_0 \qquad (18b)$$

EE contacts are tracked by the geometry module. This is necessary because the contact location can occur anywhere along the edges involved, and the dynamics module has no information about the internal geometry of objects. As in the VF case, the two edges involved are reported in the contact-block generated when the contact is first detected. Subsequently, as long as the contact is in effect, the geometry module determines the location on each edge that is closest to the other edge. The point halfway between these two locations is taken as the point of contact. The contact normal is formed by taking the cross product of the two edge direction vectors. The signs are selected so that each normal points away from its object.

4.4.3. *Special Cases during Contact Analysis*

Special situations arise during contact analysis that have to be recognized and afforded special treatment. They occur for two reasons: firstly, because of the interpolation technique used to determine initial contact locations; and secondly, because of the polyhedral representation used, which can result in exceptional relationships between the vertices and edges of pairs of objects.

Virtual contacts due to interpolation. *Virtual* contacts are contacts that must be included in the contact state even though there is actually no interference between the objects involved. They can arise when the location of an initial contact is determined by linear interpolation, as shown in Fig. 9a. The system is integrated forward by the current integration time step, dt, from t_1 to t_2. At t_1, with the vertex at $\mathbf{v}_1$, there is no interference between objects **A** and **B**. At t_2, with the vertex at $\mathbf{v}_2$, there is interference. Linear interpolation predicts motion along the dotted line, resulting in a predicted initial contact at time $t_c (t_1 \le t_c \le t_2)$. The equations of motion are now integrated forward to t_c,

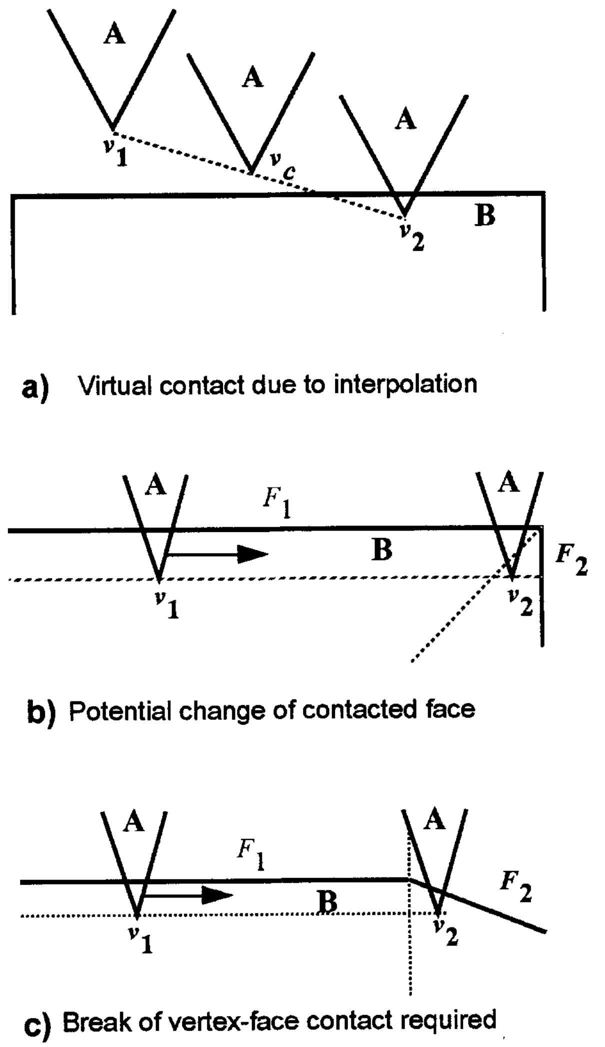

A
A
A
v_1
v_c
v_2
B
a) Virtual contact due to interpolation
A
F_1
A
B
F_2
v_1
v_2
b) Potential change of contacted face
A
F_1
A
F_2
B
v_1
v_2
c) Break of vertex-face contact required

Fig. 9. Special contact analysis cases.

with the vertex reaching position $\mathbf{v}_c$. It is possible, however, that we have backed up too far, and no interference occurs at t_c, as shown in Fig. 9a. Nonetheless, we assume that the one-step interpolation technique achieved sufficient accuracy and include the contact as a *new* contact in the contact-state, $\mathbf{C}_c$, at time t_c. The contact is treated as if initial contact occurred at the interpolated position. The virtual contact remains in the contact-state at subsequent times as a *continuing* contact as long as the spring–damper force for the contact remains positive (compressive).

Maintaining face continuity in a VF contact Consider a *continuing* contact in which vertex $\mathbf{v}$ of object $\mathbf{A}$ slides along face F_1 of object $\mathbf{B}$. As shown in Fig. 9b, the penetrating vertex is located at $\mathbf{v}_1$ at time t_1, and moves along the dotted line to location $\mathbf{v}_2$ at time t_2. At t_1, $\mathbf{v}$ is unambiguously closest to face F_1. If the sliding persists long enough, the vertex will eventually cross the plane that bisects the angle between faces F_1 and F_2 (represented by the dotted angle bisector), reaching a location such as $\mathbf{v}_2$. At this point, $\mathbf{v}$ is closer to F_2 than to F_1, which would normally be reported as a contact between $\mathbf{v}$ and F_2.

This error is avoided by requiring that a contacting vertex remain in contact with the same face of the contacted object for the entire duration of a continuing contact. Whenever a vertex $\mathbf{v}$ is found to penetrate an object in a new dynamic state, the previously accepted contact-state list is scanned for an occurrence of a contact involving $\mathbf{v}$. If the vertex is found, the object it is in contact with is determined. If the old contacted object is the same as the new contacted object, the new penetrated face is taken to be the same face that was penetrated in the old contact.

Breaking contacts when an edge is crossed Normally, a contact is broken when the dynamics module reports a zero or negative (tensile) inter-object normal contact force. A negative force would imply an attractive rather than repulsive, interaction, which is a physical impossibility. For continuing contacts, however, a geometric condition requiring that the contact be broken may occur before the contact force condition is satisfied. Specifically, for a VF contact, the vertex crossing an edge of the contacted face's polygonal boundary implies that it is no longer in contact with the face. Figure 9c shows how this happens for a VF contact. Vertex $\mathbf{v}$ of object $\mathbf{A}$ is stably sliding along face F_1 of object $\mathbf{B}$, starting at location $\mathbf{v}_1$ at time t_1. It has penetrated $\mathbf{B}$ to the depth of the dotted line, thereby producing an upward contact force that exactly balances the downward gravity force. At time t_2 it reaches the location $\mathbf{v}_2$, which is beyond the perimeter of F_1's face polygon. Clearly, $\mathbf{v}$ can no longer be in contact with F_1. Therefore, a VF contact is broken for geometric reasons if the contacting vertex, when projected normally onto the plane containing the contacted face, does not lie inside the face polygon.

A similar geometric condition may have to be used to break an EE contact for geometric reasons. Consider each edge to be a finite segment along an infinite line vector. Next consider the line segment that joins the two infinite lines that is mutually perpendicular to both of them (i.e. it joins the closest point on each of the infinite lines). It interesects the infinite line segments containing edges E_1 and E_2 at points $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively. The geometric condition for breaking the EE contact is that $\mathbf{u}_1$ lies outside the finite extent of edge E_1, or that $\mathbf{u}_2$ lies outside the finite extent of edge E_2.

5. Simulation System Environment

The simulation system is implemented on a Silicon Graphics Power Series® 4D/340 system with a 33 MHz MIPS3000® microprocessor for standard mathematical computing, and high performance custom VLSI processors for graphics processing. The software is implemented in C and runs in a standard Unix® environment.

The program is initialized from an input file that contains the simulated objects' geometries, their initial positions and velocities, material properties, and other parameters that control the simulation such as time step, integration error tolerance, etc. Unspecified parameters receive default values.

5.1. Input File and Program Initialization

The essential contents of the input file are object geometries. An object's geometry is defined by object definition primitives, such as *cube* or *prism*. Objects are defined in a body centered coordinate system whose origin is at the object's geometric center. The axis of symmetry for *prism* type objects is always the z-axis. A primitive objective has a 1 meter extent along the x and z axes, and a known extent (exactly or approximately 1 meter, depending on the specific object definition) along the y-axis. Additional parameters of the object primitive specify three scale factors that size the object along the x, y and z axes. For example, the command

```
prism  block red  6  0.25 0.50 0.75
```

defines a right rectangular prism named 'block' whose color is red. The bottom and top faces are polygons orthogonal to the z-axis. The first of the four numerical parameters specifies the number of edges, 6 in this case, in the bottom and top polygons. The next three parameters are x, y and z scale factors for sizing the object. Each object's initial position and velocity are specified using the *loc* and *vel* commands. For example

```
loc  x y z  θ u v w
```

positions the body with its center of mass at (x, y, z) in the world coordinate system, and rotates the body by θ degrees about the vector (u, v, w), and

```
vel  x y z  θ u v w
```

gives the body an initial linear velocity of (x, y, z) (meters/second) and an initial angular velocity of θ (degrees/second) about the body fixed axis (u, v, w).

Inertial properties are computed automatically from the shape and dimensions specified for each object. Objects are assumed to be of uniform density which, by default, is that of water (10^3 kg/m^3). A different density can be set using the *spgrav* command. Exact values for inertial properties are computed for non-tapered polyhedra. For tapered objects, an approximation is used to simplify the calculation. The inertial properties computed are those of the truncated elliptical cone that circumscribes the defined polyhedron and has the same height (z dimension) as the tapered prism.

An object's material properties affect the contact forces that occur when it interacts with other objects. The *mu* command sets a body's base coefficient of friction. Since the coefficient of friction used in Coulomb's law really applied to an interaction between two bodies, the actual μ used at a contact is the geometric mean of the two bodies' individual base coefficients of friction. The *kn*, *kt*, *cn* and *ct* commands set, respectively, the normal and tangential spring constants and the normal and tangential damper coefficients associated with an object's surface elements. Integration control parameters specified in the input control file include an error tolerance parameter, an initial time step, and minimum and maximum time step limits. During initialization, the minimum time step is checked against the time constants of the object's surface elements. If necessary, appropriate changes are automatically made to the object's material parameters to make the surface element time constant sufficiently larger than the minimum integration step. During the simulation, an adaptive fourth-order Runga–Kutta integration scheme controls the time step.

5.2. Interactive Visualization

Results are visualized through an interactive animated display on the CRT of the simulation workstation. The animation can be automatically recorded on video tape, if desired. The visualization is based on a system that was developed previously for displaying simulated robot manipulators [17, 18]. Objects can be displayed with solid colored surfaces illuminated by a Phong lighting model [19] or, optionally, as wireframe models.

Results can be visualized either during simulation or later in a *movie playback* mode. In the latter case one can control the rate of frame display to play at various speeds, for example in slow motion.

6. System Performance

Ideally, we would like simulations to run in 'real-time', i.e. at the same rate that events occur in the physical world. The integration time steps we use range from about 0.01 ms (when objects are in contact) to 1 ms (when there are no contacts). This means that for real-time performance the system must perform calculations for 1000 to 100,000 integration steps per second, including equation integration, contact analysis and image rendering. Not surprisingly, real-time performance is not achieved even for simulations involving small numbers of objects (e.g. 10 or fewer moving objects and a comparable number of fixed objects). Computation time to real-time ratios are, rather, observed to be between 20:1 and 500:1 for such simulations, depending on details of the simulation.

Equation integration and 3-D contact analysis of this complexity would not have been feasible a decade ago on anything but the highest performance supercomputers. Fortunately, sufficient performance for such computations is available today using high performance workstations.

Simulation performance is difficult to characterize simply, but three factors are dominant:

N	the total number of moving vertices;
τ	the fraction of total simulated time that objects are in contact; and
α	the number of active contacts at any time.

As we saw earlier, interference detection computations at each time step grow as N^2. The hierarchical test strategy used makes the computation extremely fast for well separated objects. In crowded environments (which are often of most interest), however, the detailed interference detection algorithm must be used. Additional fixed objects also increase computation. However, their effect grows only linearly, so they are inherently less of an issue than moving objects.

τ is important for two reasons. Firstly, because the integration time step while any contact exists is typically an order of magnitude or more smaller than when no contacts are in effect. All other things being equal, this slows progress by at least a factor of 10. Secondly, the most complex geometric computation, contact analysis, must be performed while objects are in contact.

α is significant because it determines the amount of contact analysis computation that must be performed. When objects are bouncing off each other and impact is the dominant contact mode there are few simultaneous contacts. However, for simulations with energy loss, bodies eventually stop moving and come to rest on some supporting surface. Toward the end of such a simulation, many bodies have multiple continuing contacts with the supporting surface. Even though nothing much is happening, all of these contacts must be tracked, slowing the simulation considerably. When τ is substantial and at the same time there are many active contacts, the situation is doubly difficult. As contact activity gets large, more and more computation goes into contact analysis.

7. Conclusions

We have successfully implemented a discrete version of the contact model proposed in [1] in a software system for simulating the dynamics of freely interacting rigid bodies, including friction. Such 'contact mechanics oriented simulation systems' can be applied in areas such as CAD/CAM, automation, rapid prototyping, animation, biomechanics, education and 'virtual reality'. Computing contact mechanics is difficult because of the complexity of material behavior and the detailed geometric calculations required to determine contact forces.

We estimate contact forces by modeling localized material deformation in the vicinity of contact. Implementation of the contact model involved the formulation of its discrete version, tuning of model parameters to reflect the essence of material behavior, control of these parameters to achieve stable and meaningful numerical integration and construction of a geometry analysis subsystem to deal with issues involved in contact analysis. We tested our system on several simulations that provide strong visual validation on its 'correctness'. Additionally, it performed very well on the *Gedanken* experiments detailed in [1].

We were interested in the quality of the simulation results, in terms of the naturalness and correctness of the resulting motions, and in understanding the difficulties that might arise computationally from integrating stiff differential equations that change at discrete contact events. The results on both counts have been positive. Simulated motions are extremely realistic. Even with the current single processor workstation implementation, performance is reasonable for modest numbers of objects, requiring 20 to 500 times real-time to obtain results. The interactive visualization capabilities provide a very convenient user interface with great flexibility for focusing on particular objects or particular areas of space of interest. With the movie playback capability, real-time visualization is easily obtained, as is slow motion or fast motion playback.

Further work on system optimization could lead to improved performance in several areas. Interference detection and contact analysis algorithm improvements are possible, such as partitioning space to limit the number of interfering object pairs that must be considered at any time. Also, available information about active contact elements (i.e. vertices, faces and edges) from the previous contact-state can be used to guide the computation for the current contact-state, a technique called *temporal coherence*. Research that would lead to linear or $n \cdot \log(n)$, rather than n^2, complexity in the interference and contact analysis algorithms would bring major improvements. Finally, adding multiprocessing capability to the software would have direct and substantial benefits. The nature of the problem and of the simulator design is such that multiprocessing is a readily achievable objective. Many computations involve identical computations that are applied to different faces, edges and vertices. These could be parallelized with modest effort and substantial pay-off. The dynamics integration program could obtain similar advantage by applying different processors to different objects.

Acknowledgements

Thanks are again given to all those acknowledged in Part I of this paper. We would also like to extend our gratitude for helpful discussions with: David Kapilow on 'C' and the 'Pi debugger', Win Nelson on damper constant issues, and Mike Brown on numerical issues.

References

1. Goyal, S.; Pinson, E.N.; Sinden, F.W. (1994) Simulation of dynamics of interacting rigid bodies including friction I: general problem and contact model, Engineering with Computers, 10, 161–173.
2. Routh, E.J. (1960) Dynamics of a System of Rigid Bodies, 7th edn, Dover Publications, New York (first published by Macmillan, London, 1860)
3. Goldsmith, W. (1960) Impact: The Theory and Physical Behaviour of Colliding Solids, Edward Arnold, London
4. Hahn, J.K. (1988) Realistic animation of rigid bodies, Computer Graphics, 22, 4, 299–306
5. Armstrong, W.W.; Green, M.W. (1985) The dynamics of articulated rigid bodies for purposes of animation, Proc. Graphics Interface, 85, 407–415
6. Baraff, D. (1989) Analytical methods for dynamic simulation of non-penetrating rigid bodies, Computer Graphics, 23, 3, 223–232
7. Walton, O.R.; Maddix, D.M.; Butkovich, T.R.; Heuzé, F.E. (1991) Redirection of dynamic compressive waves in materials with nearly orthogonal and random joint sets, Proc. ASME Applied Mech. Conf., Recent Advances in Mech. of Structured Continua, June
8. Cundall, P.A. (1988) Formulation of a three-dimensional distinct element model – Part I: a scheme to detect and represent contacts in a system composed of many polyhedral blocks, Int. J. Rock Mechanics, 25, 107–116
9. Cundall, P.A.; Strack, O.D. (1979) 'A discrete numerical model for granular assemblies, Geotechnique, 29, 47–65
10. Goyal, S. (1989) Second order kinematic constraint between two bodies rolling, twisting and slipping against each other while maintaining point contact, TR 89-1043, Department of Computer Science, Cornell University, Ithaca, New York
11. Brenan, K.E.; Campbell, S.L.; Petzold, L.R. (1989) Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, Elsevier, New York
12. Brown, M.K. (1985) A controlled impedance robot gripper, AT&T Technical Journal, 64, 4, 937–969
13. Fung, Y.C. (1965) Foundations of solid Mechanics, Prentice-Hall, Englewood Cliffs, New Jersey
14. Brach, R.M. (1991) Mechanical Impact Dynamics, Wiley, New York
15. Meirovitch, L. (1986) Elements of Vibration Analysis, McGraw-Hill, New York
16. Pinson, E.N. (1987) A software library for collision detection calculations, AT&T Bell Labs Internal Memorandum, New Jersey, October.
17. Pinson, E.N. (1985) A simulation environment for robot software development, Proc. 4th Cambridge Symposium on Intelligent Robots and Computer Vision, September, pp. 270–276
18. Pinson, E.N. (1986) Modeling and simulation for robot software development, Proc. Japan–U.S.A. Symposium on Flexible Automation, Osaka, Japan, July, 614–619
19. Phong, B.T. (1975) Illumination for computer generated pictures, CACM, 18, 6 (June), Cambridge, Massachusetts, 311–317

Appendix A: Effective Mass during Off-centric Impact

The effective mass of a body along a certain direction, say $\mathbf{n}$, at a contact point C, is the point mass m_{eff} whose acceleration along $\mathbf{n}$ is identical to that of body fixed point C when they are both (body and point mass) subjected to the same force at C.

We shall derive the expression for m_{eff} during impact for the body shown in Fig. 10. All non-scalar quantities are expressed with reference to the body fixed xyz coordinate system with its origin at the CM (center of mass) at O. Let $\mathbf{J}$ be the inertia tensor for the body, m its mass, α and ω its angular acceleration and velocity respectively, $\mathbf{a}_o$ and $\mathbf{v}_o$ the acceleration and velocity of its CM respectively, and $\mathbf{f}_c$ the contact force at $\mathbf{r}_c$. Considering $\alpha \gg \omega$ during impact, the equations of motion for the body are $\mathbf{f}_c = m\mathbf{a}_o$, $\mathbf{J}\alpha = r_c \times \mathbf{f}_c$, and the acceleration $\mathbf{a}_c$ of point C is given by $\mathbf{a}_c = \mathbf{a}_o + \alpha \times \mathbf{r}_c$. With $\mathbf{I}$ as the identity matrix, these equations can be combined to give:

$$\begin{aligned}\mathbf{a}_c &= \frac{\mathbf{f}_c}{m} + (\mathbf{J}^{-1}(\mathbf{r}_c \times \mathbf{f}_c)) \times \mathbf{r}_c \\ &= \left(\left(\frac{1}{m} + \mathbf{r}_c \cdot \mathbf{r}_c \operatorname{tr}(\mathbf{J}^{-1}) - \mathbf{r}_c^T \mathbf{J}^{-1} \mathbf{r}_c\right)\mathbf{I} + \mathbf{J}^{-1}\mathbf{r}_c\mathbf{r}_c^T + \mathbf{r}_c\mathbf{r}_c^T\mathbf{J}^{-1}\right. \\ &\qquad \left. - \operatorname{tr}(\mathbf{J}^{-1})\mathbf{r}_c\mathbf{r}_c^T - (\mathbf{r}_c \cdot \mathbf{r}_c)\mathbf{J}^{-1}\right)\mathbf{f}_c \end{aligned} \tag{19}$$

If $a_{cn} = \mathbf{a}_c \cdot \mathbf{n}$ and $\mathbf{f}_c = f_{cn}\mathbf{n}$ then the effective mass can be obtained from:

$$m_{\text{eff}} a_{cn} = f_{cn} \quad \Rightarrow \quad m_{\text{eff}} = \frac{1}{\mathbf{n}^T \mathbf{M} \mathbf{n}} \tag{20}$$

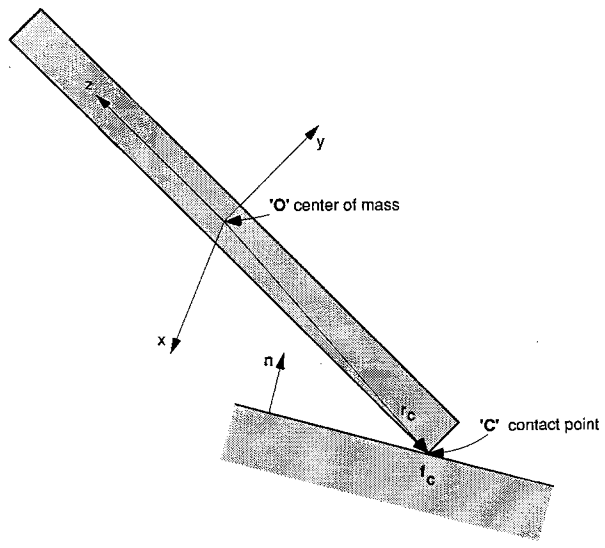


Fig. 10. Estimating effective mass of body at point C along **n**.

Appendix B: Estimate of Minimum Effective Mass

Given the geometry and the principal inertias of a convex homogenous body, as in Fig. 10, equation (20) allows us to estimate the minimum effective mass $m_{\min}$ that the body could have at a point on its surface. Let I_x, I_y, I_z be the principal mass moments of inertia, $\mathbf{r}_c = [r_x, r_y, r_z]$ and $\mathbf{n} = [n_x, n_y, n_z]$. Then

$$\frac{1}{m_{\text{eff}}} = \frac{1}{m} + \frac{(n_y r_z - n_z r_y)^2}{I_x} + \frac{(n_x r_z - n_z r_x)^2}{I_y} + \frac{(n_x r_y - n_y r_x)^2}{I_z} \tag{21}$$

Note that $m_{\min}$ is the minimum m_{eff} that can be obtained from equation (21). So $m_{\min}$ can be obtained by maximizing the right-hand side (R.H.S.) of equation (21). An approximation can be found as follows. If **n** is any one of the principal directions, then two out of its three scalar components are zero and the R.H.S. of equation (21) involves two scalar components of $\mathbf{r}_c$ (that have to be maximized) and two principal inertias (that have to be minimized).

Now let us consider the example of a body with $I_x \geq I_y \geq I_z$. The point furthest away from O is most likely to have $r_z \geq r_y \geq r_x$. Equation (21) shows that for this body, $m_{\min}$ is attained along the direction $\mathbf{n} = [1, 0, 0]$ and is given as

$$m_{\min} = \frac{m I_y I_z}{I_y I_z + m(r_y^2 I_y + r_z^2 I_z)} \tag{22}$$

In words, equation (22) implies that the minimum effective mass is felt along the direction of maximum inertia and at the furthermost point from the axis of maximum inertia.

Appendix C: Determining Initial Contact through Interpolation

The time and location of initial contacts are determined by interpolation (see Fig. 11). We know the positions of objects A and B at times t_1 and t_2, the start and proposed end of an integration interval, respectively. Therefore, $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{v}_1$ and $\mathbf{v}_2$ are known. Furthermore, we know the positions of a specific point on the penetrated face, say one of the vertices of the face polygon. Let this point be located at $\mathbf{u}_1$ at time t_1, and at $\mathbf{u}_2$ at time t_2. We assume that **v** and **u** translate linearly, and **n** rotates linearly between t_1 and t_2, a reasonable approximation if the time interval $t_2 - t_1$ is small. Then **v**, **n** and **u** can each be expressed in terms of a linear interpolation parameter t $(0 \leq t \leq 1)$:

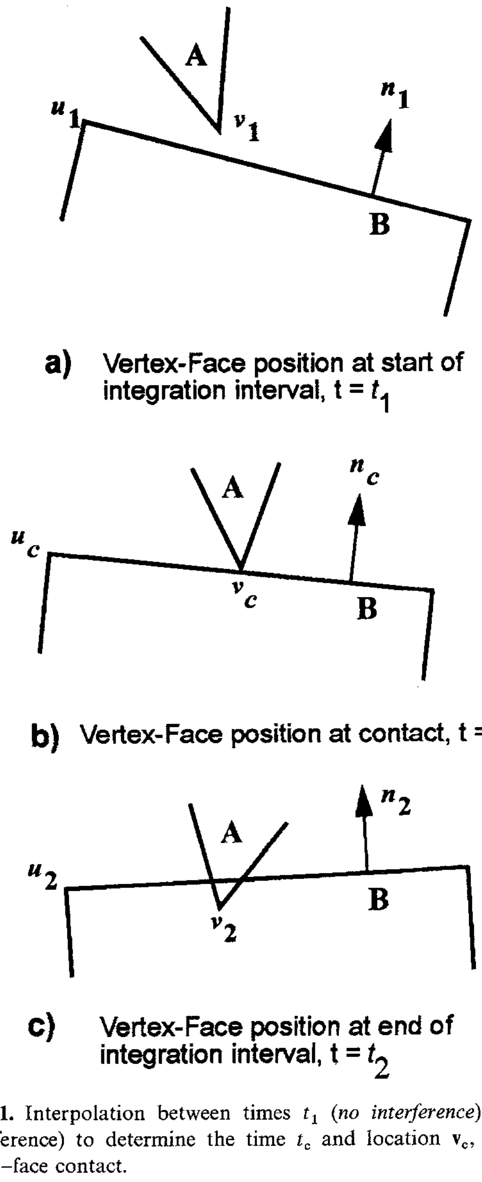
$$\left.\begin{aligned} \mathbf{v} = (1-t)\mathbf{v}_1 + t\mathbf{v}_2, \qquad \mathbf{n} = (1-t)\mathbf{n}_1 + t\mathbf{n}_2, \\ \mathbf{u} = (1-t)\mathbf{u}_1 + t\mathbf{u}_2 \end{aligned}\right\} \tag{23}$$

The equation stating that at contact the vertex must lie exactly on the face is $\mathbf{n}\cdot\mathbf{y} = \mathbf{n}\cdot\mathbf{u}$. Substituting from (23) leads to the following quadratic equation in t:

$$at^2 + bt + c = 0 \tag{24}$$

where

$$\begin{aligned} a &= (\mathbf{n}_2 - \mathbf{n}_1)\cdot[(\mathbf{v}_2 - \mathbf{v}_1) - (\mathbf{u}_2 - \mathbf{u}_1)] \\ b &= (\mathbf{n}_2 - \mathbf{n}_1)\cdot(\mathbf{v}_1 - \mathbf{u}_1) + \mathbf{n}_1\cdot[(\mathbf{v}_2 - \mathbf{v}_1) - (\mathbf{u}_2 - \mathbf{u}_1)] \\ c &= \mathbf{n}_1\cdot(\mathbf{v}_1 - \mathbf{u}_1) \end{aligned} \tag{25}$$



A
u1
v1
n1
B
a) Vertex-Face position at start of integration interval, t = t1
A
nc
uc
vc
B
b) Vertex-Face position at contact, t = tc
A
n2
u2
B
v2
c) Vertex-Face position at end of integration interval, t = t2

Fig. 11. Interpolation between times t_1 (*no interference*) *and* t_2 (interference) to determine the time t_c and location $\mathbf{v}_c$, $\mathbf{n}_c$ of a vertex–face contact.

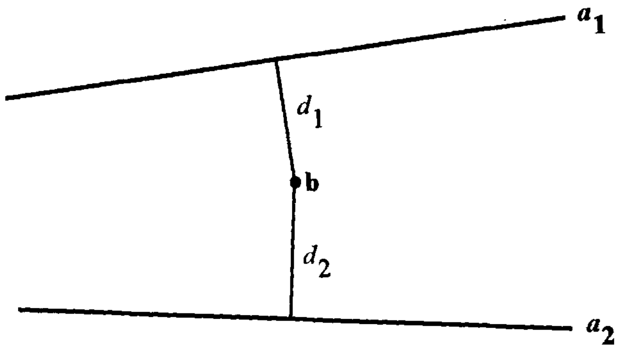


Fig. 12. Edge–edge contact determination, viewed along edge **b**, which is fixed in this coordinate system.

Equation (24) is easily solved for t. When the time interval $t_2 - t_1$ is small, exactly one of the two solutions lies in the range $0 \le t \le 1$. Note that if the penetrated body does not rotate $\mathbf{n}_2 = \mathbf{n}_1$, so $a = 0$ and (24) reduces to an even more easily solved linear equation in t.

Interpolation is also used to determine the time and location of EE contacts. The computation is performed in a coordinate system in which one object, say object **B**, remains fixed, and only object **A** moves. After solving for the time and location of initial contact in this coordinate system, the location is transformed to its proper value in the world coordinate system. The method used homogeneous transforms (HTs) as explained below.

HTs are used in the geometry module to specify object locations and orientations. An HT is a 4×4 matrix which includes a 3×3 rotation matrix and a 3×1 spatial displacement. The effect of applying (pre-multiplying) the transform to any point is to rotate the point about the coordinate system origin as specified by the rotation matrix, followed by translating the point as specified by the spatial displacement. Each object is initially defined in its own local reference frame whose origin is at the object's geometric center. At every integration time step, an HT is known for each object that takes it to its correct spatial location. In terms of HTs, the following method converts the location of object **A** from its location before interference to its equivalent location in a coordinate system where object **B** remains fixed at its post-interference location.

Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be HTs that position object **A** just before (time t_1) and just after (time t_2) interfering with object **B**, respectively. Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be HTs that position object **B** at times t_1 and t_2, respectively. Then $\mathbf{S}_2$ is used to position **B**, which remains fixed; $\mathbf{R}_2$ is used to position **A** at t_2; and $\mathbf{R}_0 = \mathbf{S}_2\mathbf{S}_1^{-1}\mathbf{R}_1$ is used to position **A** at t_1.

The computation in the fixed coordinate system is illustrated in Fig. 12, where edge **b** is the fixed edge of object **B**, while edge **a** of object A moves from location $\mathbf{a}_1$ at time t_1 to $\mathbf{a}_2$ at time t_2. Again, it is assumed that motions during this small time interval are linear (i.e. uniform translation and rotation during the time interal). Since the equations for edge **a** are known at times t_1 and t_2, it is easy to solve for d_1 and d_2, the distances between **a** and **b** at t_1 and t_2, respectively. It is also straightforward to compute $\mathbf{p}_1$ and $\mathbf{p}_2$, the points on **b** that are closest to $\mathbf{a}_1$ and $\mathbf{a}_2$, respectively. The time of contact, t_c, and point of contact on **b**, $\mathbf{p}_c$, are then determined by linear interpolation to be:

$$t_c = \frac{d_1}{d_1 + d_2}, \qquad \mathbf{p}_c = (1 - t_c)\mathbf{p}_1 + \mathbf{t}_c\mathbf{p}_2 \tag{26}$$

Knowing t_c, linear interpolation can be applied to the HTs to obtain $\mathbf{R}_c$ and $\mathbf{S}_c$, the transforms for A and B, respectively, at the time of contact. Linear interpolation between two HTs involves independent interpolation of the displacement and rotation components. Assume the initial and end times are normalized to 0 and 1, respectively. Let $\mathbf{x}_0$ and $\mathbf{Q}_0$ be the displacement and rotation components at $t = 0$, $\mathbf{x}_1$ and $\mathbf{Q}_1$ be the displacement and rotation components at $t = 1$, $\mathbf{x}_c$ and $\mathbf{Q}_c$ be the interpolated displacement and rotation components at the time of contact t_c $(0 \le t_c \le 1)$, and $\mathbf{q}$ be the incremental rotation between $\mathbf{Q}_0$ and $\mathbf{Q}_1$, i.e. $\mathbf{q} = \mathbf{Q}_1\mathbf{Q}_0^{-1}$.

For interpolation purposes, we interpret $\mathbf{q}$ to be a rotation by an angle θ about a fixed axis $\mathbf{v}$, which we can express as $\mathbf{rot}(\mathbf{v}, \theta)$. The displacement $\mathbf{x}_c$ and rotation $\mathbf{q}_c$ at contact are just

$$\mathbf{x}_c = (1 - t_c)\mathbf{x}_0 + t_c\mathbf{x}_1, \qquad \mathbf{q}_c = \mathbf{rot}(\mathbf{v}, t_c\theta) \tag{27}$$