GRAPHICAL MODELS AND IMAGE PROCESSING **60,** 461–487 (1998)
ARTICLE NO. IP980484

Deformable Object Modeling Using the Time-Dependent Finite Element Method[1]

Jie Shen[2] and Yee-Hong Yang[3]

Computer Vision and Graphics Laboratory, Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada S7N 5A9

Received February 27, 1998; revised September 3, 1998; accepted September 17, 1998

Continuum mechanics and the finite element method are used as basic tools to facilitate the animation of deformable objects. Two unique features in this paper are: (1) introduction of a framework for 3-D finite interface elements to deal with object interaction and (2) a simple approach to handle fracture propagation inside objects. Animation sequences have been constructed, and they indicate that the system is able to provide a realistic depiction of dynamic collision processes between deformable objects and a simple presentation of fracture propagation inside objects.

1. INTRODUCTION

In physically based animation, the motions of objects in a system are calculated in accordance with physical laws such as Newtonian mechanics. Generally speaking, two types of objects, rigid and deformable, are encountered in the real world. The numerical procedures to compute the movement of rigid objects are well understood in the classical mechanics area and have been successfully used in animation [1, 4, 17, 20, 22, 40]. In contrast, the modeling of deformable objects and the handling of the interactions among them are still active research areas.

Terzopoulos *et al.* [38] propose one of the early methods in the animation of deformable objects. Their model is designed to describe deformable materials such as rubber, cloth, paper, and flexible metals on the basis of the theory of elasticity. In their formulation, the equations of motion governing the dynamics of deformable objects are obtained from Newtonian mechanics by considering the balance of the external forces with the forces due to deformation, dissipation, and inertia. Later, Terzopoulos and Fleischer [37] employ

[1] The authors acknowledge financial support from NSERC through Grant OGP0000370.

[2] Shen acknowledges the University of Saskatchewan for the graduate scholarship award.

[3] To whom correspondence should be addressed.

1077-3169/98 $25.00

viscoelasticity, plasticity, and fracture to model nonrecoverable deformation as well as viscous deformation.

The Terzopoulos model has been extended by Gallagher *et al.* and Gourret *et al.* [16, 18]. As well, others have simulated the movement of a skirt blowing in a breeze, the collision between cloth and human body, and the interaction among different parts of cloth [8, 24]. A similar extension was done using a distributed force model in order to more accurately simulate the response of cloth to air flow [26].

Reeves was the first to propose the particle system approach to the computer graphics community in 1983 [31]. Two years later, the concept of particle systems was extended to the modeling of trees and grass [32]. On the basis of some of these concepts, ocean waves were modeled by Fournier and Reeves [15]. Sim [33] explored the application of a particle system in producing the visual effects of waterfalls, vortex fields, fire, snowstorms, and explosions on a massively parallel computer.

Even though a considerable amount of work has been conducted in the area of deformable object modeling during the past decade, some questions remain unanswered. One is related to the interaction among deformable objects. Most previous studies focused on the modeling of individual deformable objects with less attention to dynamic interactions among them. In general, the mechanical behavior of objects at their interface with their environment or other objects is relatively more complex than that inside the objects. Another question is fracture, especially the visualization of fracture propagation inside objects.

The finite element method (FEM) is the most powerful numerical technique for the analysis of any complex system which can be discretized to a finite number of well-defined components. One main advantage over other methods (classic analytical method, particle system, and finite difference method) is its supreme capability in handling objects with any complex-shaped mesh configuration.

This paper is related to the application of the finite element method in the modeling of deformable objects. The main objectives are (1) to establish a framework of interface to simulate the interaction between objects and (2) to implement a simple scheme to simulate the fracture propagation inside an object.

The animation sequences described in this paper can be viewed at http://www.cs.usask.ca/research/research_groups/vision/projects/feanim.htm.

2. INTERACTION AMONG DEFORMABLE OBJECTS

Two steps are usually involved in simulating the interaction among deformable objects. The first step is to perform the collision detection, i.e., to detect where and when two objects collide. The second step is to determine collision response, i.e., to analyze the behavior of objects after a collision occurs.

2.1. Collision Detection

Collision detection has been extensively investigated [11]. The simplest approach is to use the bounding volume and spatial decomposition techniques. Many algorithms have been developed to address the collision detection in the modeling of cloth, dressing, and hair [8, 23, 24, 43]. In this paper, the simple detection method proposed by Moore and Wilhelms [27] is adopted. The collision between surfaces are detected by testing if any point goes through a triangular patch during each time step.

2.2. Collision Response

Since a collision process between deformable objects is usually complex, how to predict the deformation of objects during their interaction is still a challenging task. A considerate amount of effort has been put into the study of collision response in the past. The related methodologies can be grouped into the following three categories.

2.2.1. *Analytical Solution*

The analytical solution is based upon the conservation of momentum during a collision [27]. For the analysis of collision response of rigid objects, this solution is good enough because no noticeable deformation happens in the collision. Since this solution bypasses the question of determining the collision forces by calculating a new angular and linear velocity after collision according to the angular and linear velocity before collision for each object, it is not suitable for deformable objects.

2.2.2. *Constraint Methods*

One common characteristic of approaches in this category is that one or more constraints are added into the investigated system in order to simulate the real behavior of objects in a collision process. From the viewpoint of implementation, the previous approaches generally fit into one of the following.

(1) Physically inserted interface component. The primary common feature of this group is that a kind of interface component is inserted at a collision point to prevent the interpenetration between colliding objects. Investigators have proposed several different interface components.

Spring. Inserting a spring dynamically (i.e., during a simulation) at a collision point is the most intuitive way to handle collisions [27]. The precondition for using a spring is that the simulation system should accommodate the spring forces as external forces to objects. Whenever a collision occurs, a spring is temporarily inserted at the collision point. The stiffness of the spring should be large enough to withstand the impact between the two objects. After the collision ends, the spring is eliminated from the system. This constraint is easy to understand and implement.

One major problem with this approach is that it is conceptually supposed to pass only the force along the axis of the spring without considering the forces perpendicular to it. When sliding exists at the interface between two objects, the skewed spring hardly functions properly. Another problem is the connection between the spring and the surface node on the surface of each colliding object. Most often, the surface nodes on the two colliding objects are not contiguous to each other, as illustrated in Fig. 1a. One possible solution is that a transition layer is developed which consists of the projections of the nodes of surfaces of objects A and B, as shown in Fig. 1b (Simo *et al.* [34]). One drawback associated with Simo's technique is the extra layer of nodes which is difficult to locate on curved surfaces. The third shortcoming is the high computational cost when the stiffness of the spring has to be a large magnitude [2], because it requires a small time step for accurate numerical integration.

Pinball. Pinballs are statically used (i.e., pre-allocated before a simulation) to form an interface layer for each object in the system, as shown in Fig. 2 [36]. Each pinball is embedded in one finite element at the surface of the object. Then, the collision detection is

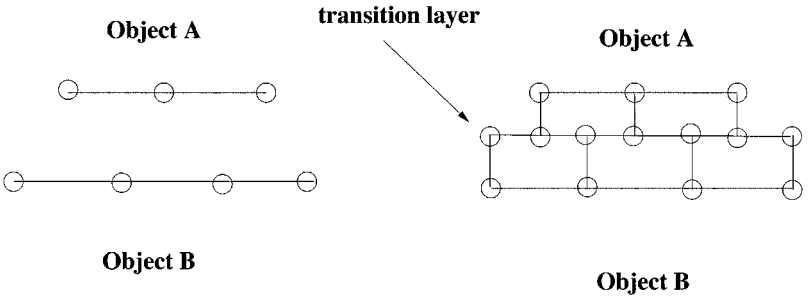

FIG. 1. Surface noncontiguous nodes and a transition layer proposed by Simo *et al.* (1985) [34].

simplified by checking only the penetration between pinballs belonging to different objects. The penetration depth is calculated by using the coordinates of the center of each pinball. On the basis of that depth, the reaction force can be calculated in a similar way by calculating the spring force. One problem with the pinball approach is that the time step before collision cannot be large. Otherwise, one ball could entirely pass another one in a single time step, resulting in no collision between these two pinballs. Another problem is that handling the friction between pinballs is still not well solved, because the sliding of two plane bodies will undoubtedly cause the oscillation of normals of pinball surfaces, which should not exist.

(2) Mathematically calculated interface force. The common feature of this group is that interface forces are calculated by using some mathematical formula which may or may not be based upon physical laws. No interface component is added to the system when a collision happens. Interface forces are treated as external forces to each object.

Arbitrary interface force. Terzopolous *et al.* [38] added a potential energy around each object to prevent the penetration between objects. The resulting force of collision is determined by the gradient of the potential. This type of approach is useful when the collision response is not an important issue in the investigated system. For instance, the

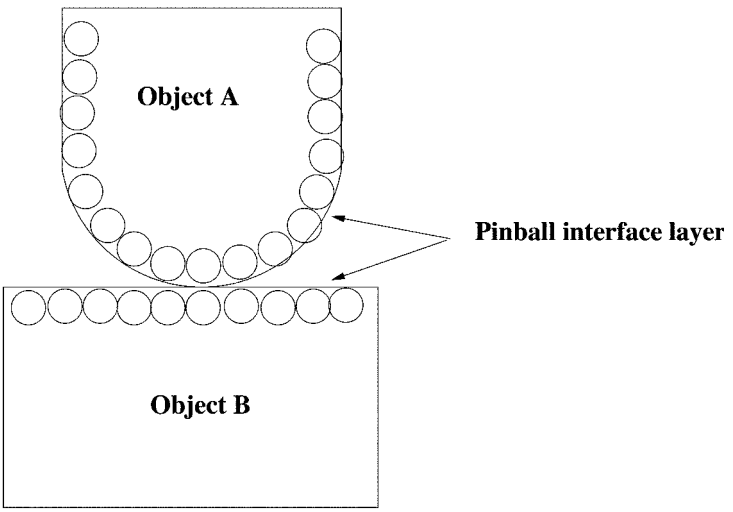

FIG. 2. A pinball interface layer shown in two dimensions.

bouncing back is not considered after a collision between two objects, i.e., two objects stick together after the collision.

Controlled path. Controlled path means that the path of movement of one or more particles of an object is specified or given before the animation process. Reaction constraints [30] are the common technique to resolve this type of problem. The basic idea of reaction constraints is to add a calculated constrained force F constrained at a given node such that the movement of this node meets the requirement of certain path constraint.

One alternative is to specify the movement of the given nodes of the object explicitly. For instance, if the displacement-based finite element method is used instead of the particle system, the displacement of those given nodes can be specified as a boundary condition to the system for each time step.

This approach is only suitable to cases where the path of some particles or nodes is known ahead of the animation process.

Constraint boundary defined by an equation. In the customary problems of classical mechanics, the equations of motion of objects can be determined by minimizing a function $f(x)$ which is a definite integral of the Lagrangian function. In some cases, the surface of the object as a constraint can be expressed by an equation, $g(x) = 0$, such as that for a rigid plane or a rigid sphere. Then, a constrained optimization procedure can be employed to find the minimum of locally minimizing the goal function, $f(x)$, subject to the constraint equation, $g(x) = 0$ [30].

As to polygenic forces of a frictional nature, they are not derivable from a work function and have no potential energy function. Consequently, they are outside the realm of variational principles, while the Newtonian scheme has no difficulty in including them [25]. For irreversible displacements, the minimization leads to a variational inequality, i.e., the Fourier's inequality.

This kind of procedure assumes that the constraint boundary can be defined by an equation $g(x) = 0$ which is valid only when the constraint boundary is known in advance. However, when two deformable objects with irregular geometric shape collide with each other, most often the contact surface is not known *a priori*. If we consider one object as the constraint to the other, it is difficult to construct the function $g(x)$ for the purpose of minimization.

Constraint boundary defined in a form of relative displacement between two objects. An alternative is to first minimize the goal function $f(x)$ to obtain the equation of motion for the system, and then to impose constraints to the equation of motion. The relative displacement and interface forces at collision points are considered as extra variables. By imposing a certain friction law, such as the Coulomb law, as a constraint, the dynamic collision between objects can be analyzed [9].

2.2.3. *Global Deformation and Local Deformation*

Witkin and Welch [42] and Baraff and Witkin [3] use a concept of global deformation to describe flexible objects. It is a compromise between the extremes of the nodal and rigid formulations. The changes in the shape of objects are approximated by the global deformations that apply a parametric "space warp" to all the discretized nodes of the objects. Since fewer nodes are used in calculating the global deformation than in the nodal approach, this scheme is fast. In addition, the stiffness problem due to local interactions is eliminated because the shape parameters are global in their effect. On the other hand, this scheme is difficult to use to simulate the complex contact surface between arbitrarily shaped flexible objects if no fine triangularization of surface patches is used for the polyhedral approximation.

Pentland and Williams [29] also achieve the goal of separate representation of dynamic behavior and geometric form by using the global deformation technique. A system is built to use polynomial deformation mappings to couple a vibration-mode ("modal") representation of object dynamics together with volumetric models of object geometry. Faloutsos *et al.* [14] extend Witkin and Welch's approach to accommodate a hierarchy of deformations which are applied to objects in a nonlinear fashion with respect to the state parameters.

Cani-Gascuel and Desbrun [7] propose an approach to handle the collision response between deformable objects by the combination of global and local deformation. An implicit isopotential surface is statically used to coat each base structure, an internal physically based model. The implicit layer performs collision detection and generates the local deformations due to contact, while the base structure controls the global scale behavior. This approach is able to generate a relatively accurate contact surface without great computation cost. However, one possible problem is that the inter-penetration region is not necessarily the area with greatest deformation during a collision. For instance, consider the situation where the inter-penetrated part of an object is much stiffer than its remaining sub-parts.

2.2.4. *Remarks*

The approaches of *physically inserted interface component* and *mathematically calculated interface force* are very similar to each other. The major difference between them is that for the former, extra elements are created with no change to the motion equations, while the latter requires an adjustment of the motion equations without the cost of extra elements. To give a realistic description of interaction between arbitrarily shaped deformable objects, both approaches need to take the cost of computation for tracking the deformed interface in some way.

Belytschko and Neal [36] indicate that if the central difference integration method is considered in a linear case, the penalty method usually causes less than 25% decrease in time step for stability analysis, and for the Lagrange multiplier method, it can be proven that there is no increase in time step by adding the Lagrangian multiplier.

Platt and Barr [30] give an excellent summation on the constraint methods on deformable objects, even though the constraints are not necessarily limited to the cases of collision response. The reaction constraints (RCs) defined by them are basically used to control the path of some nodes on an object. This is an important artistic feature for the controlling power of an animator, but is less related to the realistic presentation of collision response from a scientific point of view. The dynamic constraints based on the inverse dynamics are somewhat difficult to implement to deformable objects because more state variables are involved.

Below, a new framework for the approach of *physically inserted interface component* is introduced with an aim of providing a high-accuracy means for describing the interaction between deformable objects.

2.3. A Framework for Contact Interfaces between Objects

Adhesion and friction exist at every interface between deformable objects. In some cases, they may influence the interaction remarkably, because adhesion and external friction at an object–object contact surface are generally different from cohesion and internal friction within the objects, respectively. For an accurate finite element simulation, the interface should be simulated using one or more special types of interface element.

The interface or contact surface formed by adjacent objects can be generally classified as point-to-point, point-to-surface, surface-to-surface, point-to-edge, edge-to-edge, etc. We

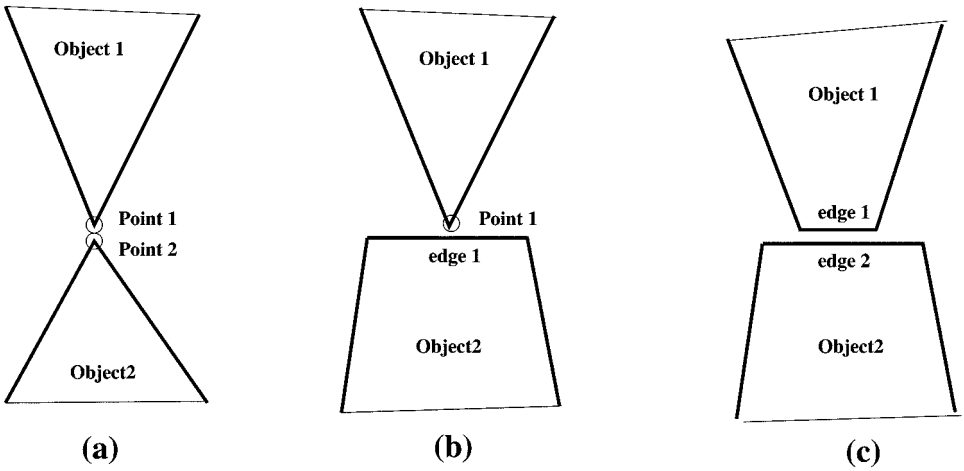

FIG. 3. Three typical types of contact interfaces between objects. (a) Point-to-point; (b) point-to-surface; (c) surface-to-surface.

can consider the point-to-edge and edge-to-edge as the special case of point-to-surface and surface-to-surface, respectively. Figure 3 shows three typical cases in two dimension. To simulate all these types of contact, the following framework is proposed, which consists of two types of interface elements.

2.3.1. *Point-to-Point Type of Interface*

A rod-like interface element is used in simulating the point-to-point interface. Such an element is called the point-to-point element which has two nodes with a length L, as shown in Fig. 4. Node i is adjacent to one object, while node j is considered as a point on the surface of the other object.

Let F_{ti}, F_{si}, and F_{ni} denote the components of total internal nodal force at node i exerted by its adjacent elements excluding the point-to-point element, and F_{tj}, F_{sj}, and F_{nj} be the total internal nodal forces at node j. Since the magnitude of L is very small, it is permissible to assume that the nodal force components at nodes i and j are equal in magnitude and opposite in direction. Therefore, for the point-to-point element with nodes i and j, six equilibrium equations in local coordinate system $t - s - n$ can be written as

$$\begin{aligned} F_{ti} &= -F_{tj}, \\ F_{si} &= -F_{sj}, \\ F_{ni} &= -F_{nj}, \\ F_{ti} &= (U_{ti} - U_{tj})k_t, \\ F_{si} &= (U_{si} - U_{sj})k_s, \quad \text{and} \\ F_{ni} &= (U_{ni} - U_{nj})k_n, \end{aligned} \tag{1}$$

where U_{ti}, U_{si}, U_{ni}, U_{tj}, U_{sj}, and U_{nj} are displacement components in directions t, s, and n at nodes i and j, respectively; k_t, k_s, and k_n are stiffness coefficients in directions t, s, and n, respectively.

The one-to-one mapping between the local coordinate system $t - s - n$ and the global coordinate system $X - Y - Z$ follows the convention below:

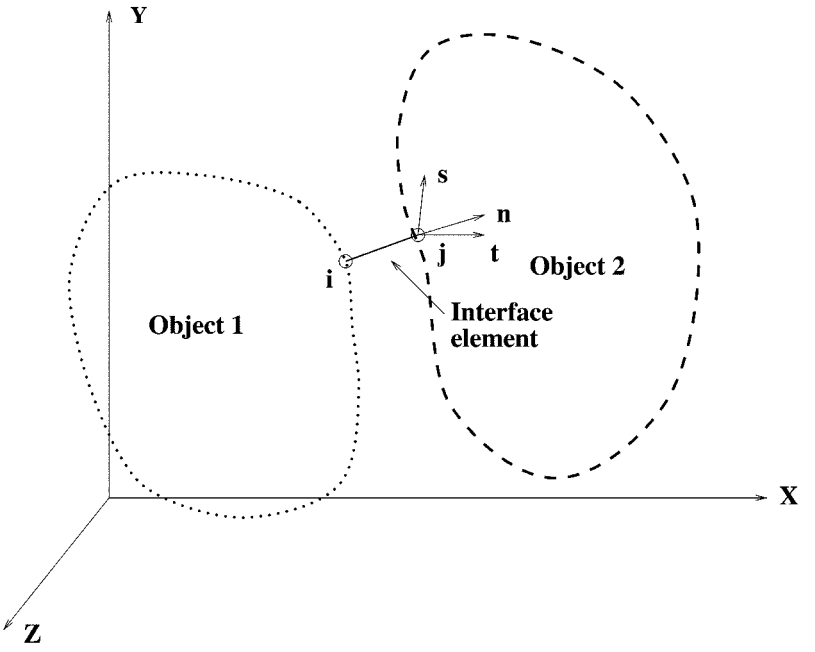

FIG. 4. A point-to-point element.

The coordinate directions in the $t-s-n$ system is obtained by rotating the $X-Y-Z$ system in two steps:

1. rotate α with respect to the Y axis, resulting in the $X'-Y'-Z'$ system
2. rotate $-\beta$ with respect to the X' axis, resulting in the $X''-Y''-Z''$ system

X'', Y'', and Z'' are in the same direction as t, s, and n, respectively, as illustrated in Fig. 5. The relationship between these two coordinate systems can be written as

$$\begin{bmatrix} dt \\ ds \\ dn \end{bmatrix} = \begin{bmatrix} t_1 & t_2 & t_3 \\ s_1 & s_2 & s_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ \sin\alpha\sin\beta & \cos\beta & \sin\beta\cos\alpha \\ \sin\alpha\cos\beta & -\sin\beta & \cos\alpha\cos\beta \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix}. \quad (2)$$

Given the coordinates of nodes i and j as (X_i, Y_i, Z_i) and (X_j, Y_j, Z_j), $\cos\alpha$, $\sin\alpha$, $\cos\beta$, and $\sin\beta$ can be easily determined. Based on the coordinate transformation in Eq. (2), the following displacement and force transformations are given, respectively, by

$$\begin{bmatrix} U_{ti} \\ U_{si} \\ U_{ni} \\ U_{tj} \\ U_{sj} \\ U_{nj} \end{bmatrix} = \begin{bmatrix} t_1 & t_2 & t_3 & 0 & 0 & 0 \\ s_1 & s_2 & s_3 & 0 & 0 & 0 \\ n_1 & n_2 & n_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_1 & t_2 & t_3 \\ 0 & 0 & 0 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & n_1 & n_2 & n_3 \end{bmatrix} \begin{bmatrix} U_{xi} \\ U_{yi} \\ U_{zi} \\ U_{xj} \\ U_{yj} \\ U_{zj} \end{bmatrix} = \mathbf{T} \begin{bmatrix} U_{xi} \\ U_{yi} \\ U_{zi} \\ U_{xj} \\ U_{yj} \\ U_{zj} \end{bmatrix}, \quad (3)$$

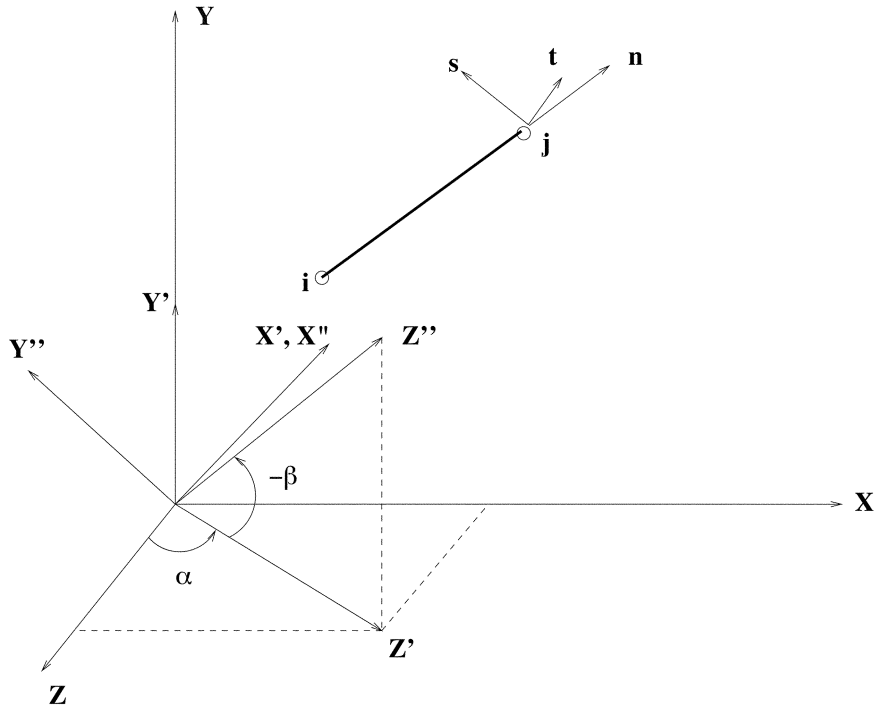

FIG. 5. Transformation from the global coordinate system $X-Y-Z$ to the local coordinate system $t-s-n$.

and

$$\begin{bmatrix} F_{ti} \\ F_{si} \\ F_{ni} \\ F_{tj} \\ F_{sj} \\ F_{nj} \end{bmatrix} = \begin{bmatrix} t_1 & t_2 & t_3 & 0 & 0 & 0 \\ s_1 & s_2 & s_3 & 0 & 0 & 0 \\ n_1 & n_2 & n_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_1 & t_2 & t_3 \\ 0 & 0 & 0 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & n_1 & n_2 & n_3 \end{bmatrix} \begin{bmatrix} F_{xi} \\ F_{yi} \\ F_{zi} \\ F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix} = \mathbf{T} \begin{bmatrix} F_{xi} \\ F_{yi} \\ F_{zi} \\ F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix}. \tag{4}$$

Equation (1) can be rewritten in matrix form

$$\begin{bmatrix} k_t & 0 & 0 & -k_t & 0 & 0 \\ 0 & k_s & 0 & 0 & -k_s & 0 \\ 0 & 0 & k_n & 0 & 0 & -k_n \\ -k_t & 0 & 0 & k_t & 0 & 0 \\ 0 & -k_s & 0 & 0 & k_s & 0 \\ 0 & 0 & -k_n & 0 & 0 & k_n \end{bmatrix} \begin{bmatrix} U_{ti} \\ U_{si} \\ U_{ni} \\ U_{tj} \\ U_{sj} \\ U_{nj} \end{bmatrix} = \begin{bmatrix} F_{ti} \\ F_{si} \\ F_{ni} \\ F_{tj} \\ F_{sj} \\ F_{nj} \end{bmatrix}. \tag{5}$$

After coordinate transformation using Eqs. (3) and (4),

$$\begin{bmatrix} F_{xi} \\ F_{yi} \\ F_{zi} \\ F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix} = \mathbf{T}^{-1} \begin{bmatrix} k_s & 0 & 0 & -k_s & 0 & 0 \\ 0 & k_t & 0 & 0 & -k_t & 0 \\ 0 & 0 & k_n & 0 & 0 & -k_n \\ -k_s & 0 & 0 & k_s & 0 & 0 \\ 0 & -k_t & 0 & 0 & k_t & 0 \\ 0 & 0 & -k_n & 0 & 0 & k_n \end{bmatrix} \mathbf{T} \begin{bmatrix} U_{xi} \\ U_{yi} \\ U_{zi} \\ U_{xj} \\ U_{yj} \\ U_{zj} \end{bmatrix}, \tag{6}$$

where $\mathbf{T}$ is an orthogonal matrix such that $\mathbf{T}^T = \mathbf{T}^{-1}$.

Therefore, the element stiffness matrix in the global coordinate system is of the form

$$\mathbf{k} = \mathbf{T}^T \begin{bmatrix} k_t & 0 & 0 & -k_t & 0 & 0 \\ 0 & k_s & 0 & 0 & -k_s & 0 \\ 0 & 0 & k_n & 0 & 0 & -k_n \\ -k_t & 0 & 0 & k_t & 0 & 0 \\ 0 & -k_s & 0 & 0 & k_s & 0 \\ 0 & 0 & -k_n & 0 & 0 & k_n \end{bmatrix} \mathbf{T}. \tag{7}$$

After each step, the displacement increment (dt, ds, dn) of a point-to-point element in the local coordinate system is calculated by

$$\begin{aligned} dt &= dt_j - dt_i, \\ ds &= ds_j - ds_i, \\ dn &= dn_j - dn_i, \end{aligned} \tag{8}$$

and

$$\begin{aligned} dt_i &= t_1\,dx_i + t_2\,dy_i + t_3\,dz_i, \\ dt_j &= t_1\,dx_j + t_2\,dy_j + t_3\,dz_j, \\ ds_i &= s_1\,dx_i + s_2\,dy_i + s_3\,dz_i, \\ ds_j &= s_1\,dx_j + s_2\,dy_j + s_3\,dz_j, \\ dn_i &= n_1\,dx_i + n_2\,dy_i + n_3\,dz_i, \\ dn_j &= n_1\,dx_j + n_2\,dy_j + n_3\,dz_j, \end{aligned} \tag{9}$$

where dt_i, ds_i, and dn_i are displacements at node i in the local coordinate system. dx_i, dy_i, and dz_i are displacements at node i in the global coordinate system.

The main application of point-to-point elements is to cover the surface of each object, as illustrated in Fig. 6 in which the length of point-to-point elements is not presented in its real scale because its magnitude would be too small to be identified. Whenever one object is created, the associated point-to-point elements are also constructed to cover its surface before any numerical analysis is conducted. This procedure is called the pre-assignment of point-to-point elements to their associated object. Another possible scheme is to delay the generation of point-to-point elements until a collision really happens between two objects and only at the actual contact area are point-to-point elements constructed. It is

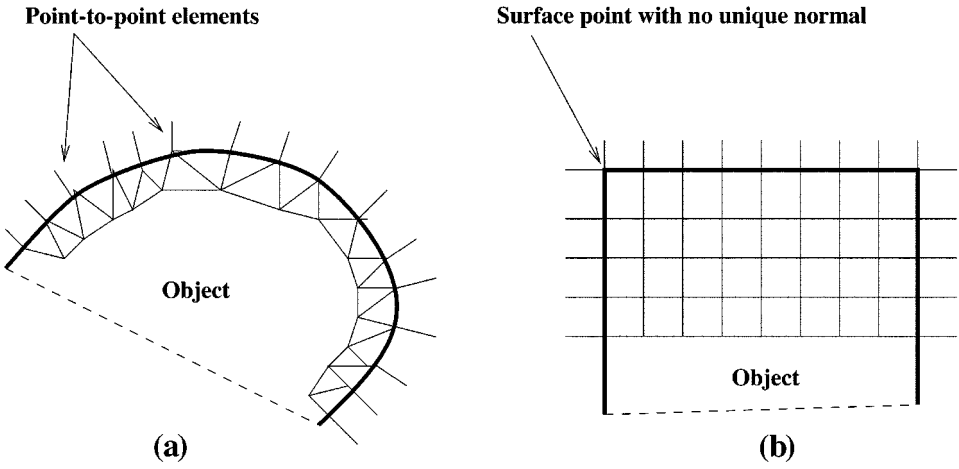

FIG. 6. Point-to-point elements used to cover the surface of objects. (a) Surface points with unique normal; (b) some surface points with nonunique normal.

called the post-assignment of point-to-point elements. The main advantage of the pre-assignment scheme is its simplicity without the need to handle problems in connection with the generation of point-to-point elements in each time step, while its major disadvantage is the waste of memory because of the pre-construction of point-to-point elements to cover the surface of the object. Compared to the pre-assignment scheme used in this paper, the post-assignment scheme has the opposite advantage and disadvantage.

At any moment, the point-to-point element may be in one of the following states:

1. Separation state. A separation state means that no point-to-point elements associated to an object touches or penetrates into other objects such that no stress and strain are developed in any elements. This state can be identified by performing a collision detection of each point-to-point element with respect to all other objects in the system.

2. Contact state. In contact state, one or more point-to-point elements touch or penetrate into surrounding objects. If one point-to-point element happens to touch one surface node of any surrounding object, this element is assumed to be fixed at that surface node such that a point-to-point contact is formed. Otherwise, a point-to-surface contact is generated, which is explained in detail next.

2.3.2. *Point-to-Surface Type of Interface*

The point-to-surface type of interface, as shown in Fig. 3, is handled by point-to-surface elements each of which is in connection with one point-to-point element, as illustrated in Fig. 7. Basically, a point-to-surface element is a tetrahedron with four nodes, three of which link to object A and the remaining one connects to a point-to-point element which in turn links to surrounding object B that touches object A. In Fig. 7, the height of the point-to-surface element, h, should be set to be quite small relative to the size of the objects. In this way, the point-to-surface element acts as an interface element to link a point-to-point element of one object to the meshed structure of the other objects.

If every three-dimensional object is divided into eight-cornered "brick" type elements, the possible connections between point-to-point and point-to-surface elements are given in Fig. 8. In this figure, d is the intersection point of a point-to-point element and its

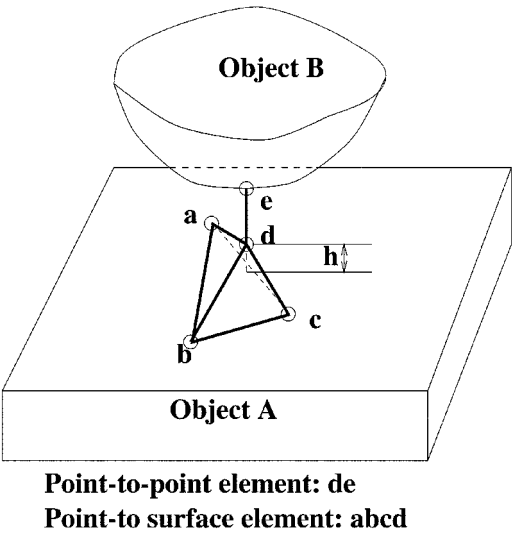

FIG. 7. Point-to-surface elements.

corresponding point-to-surface element, and overlaps with *e*, the other end of the point-to-point element. It is not necessary to assume that the position of a point-to-point element, *de*, is always perpendicular to the surface of object *A* in Fig. 7. The overlap of *d* and *e* in Fig. 8 is solely for the purpose of simplicity.

Any rectangular side of a "brick" element can be divided into two triangular elements. If *d* is within a triangular element, as shown in Fig. 8a, only one point-to-surface element is required to be generated, which provides a linkage between one node of the point-to-point element and three surface nodes on object *A*. If *d* happens to be on a diagonal line, as shown in Fig. 8b, two point-to-surface elements are required such that four surface nodes on object *A* are directly influenced by the point-to-point element in connection with *d*. Figure 8c shows a case in which *d* is located on a common boundary line of two adjacent rectangular

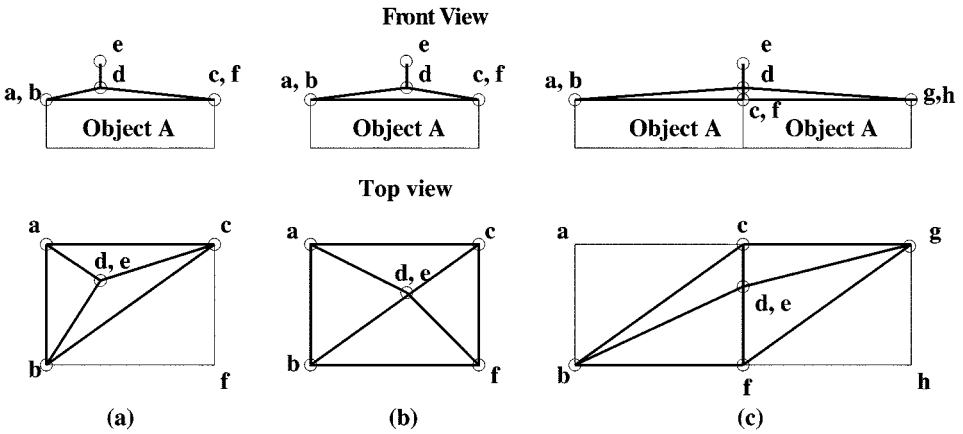

FIG. 8. Possible connections of point-to-point and point-to-surface elements. (a) Normal case (element *abcd*); (b) *d* is on a diagonal line of a rectangular side of a 'brick' (*d* is the intersection point of a point-to-point element and its corresponding point-to-surface element) (element *abcd*, *bcfd*); (c) *d* is on a boundary line of a rectangular side of a 'brick' (element *bcfd*, *cfgd*).

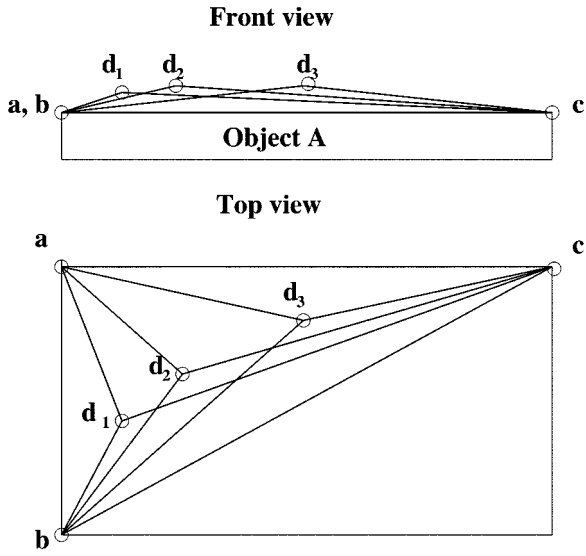

FIG. 9. One triangular surface element in connection with multiple point-to-surface elements (Point-to-surface element: $abcd_1$, $abcd_2$, and $abcd_3$).

sides. Depending on how a rectangular side is divided into two triangular patches, one pair of elements is $bcfd$ and $cfgd$, while the other pair is $acfd$ and $chfd$. It is the responsibility of an implementation to follow one type of division. The choice of two types of division in general does not matter if the mesh of objects is fine enough.

If the front end point, d, of n point-to-point elements happens to be within the same triangular surface element in Fig. 8a, n point-to-surface elements are required, as shown in Fig. 9. Even though some spatial overlaps exist among these n point-to-surface elements, there is no interference of the displacement of $d_1, d_2, \ldots, d_n$. The sole correlation among these n point-to-surface elements is that all of them are connected to the same set of surface nodes, (a, b, c), through which the forces in object B are passed to object A, and vice versa.

2.3.3. *Surface-to-Surface Type of Interface*

The combination of point-to-point and point-to-surface elements is powerful enough to handle the surface-to-surface type of interface. Figure 10 shows a typical case of the surface-to-surface type of interface. Any complex contact surfaces between objects are simulated by using the following control mechanism for the birth and death of point-to-surface elements:

- Birth control. Whenever a point-to-point element of one object falls into a triangular surface element of another object, a point-to-surface element is generated. These two elements form a pair which links the two objects.
- Death control. For a pair of point-to-point and point-to-surface elements, if the stress inside the point-to-point element satisfies its failure criteria, then this pair is broken; i.e., the point-to-surface element is eliminated and the stress in the point-to-point element is restored to zero.

The combined usage of point-to-point and point-to-surface elements can avoid the shortcoming associated with the *spring approach* when the surface nodes on the two colliding objects are not contiguous to each other. It also avoids the requirement for small time step

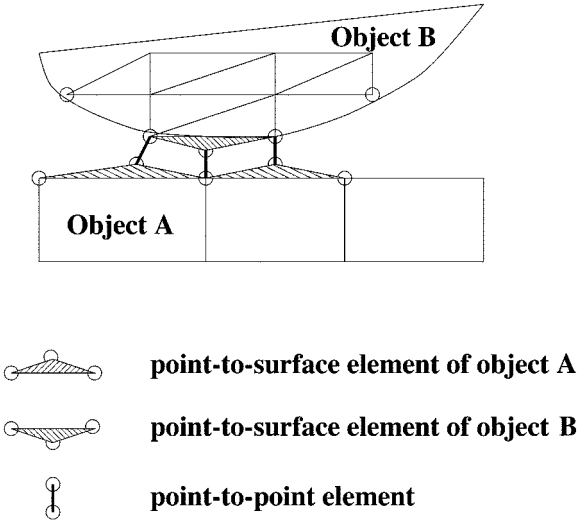

FIG. 10. A surface-to-surface type of interface.

with the *pinball approach*. The frictional and tensile failure at the interface can be handled by the death of old interface elements. After each time step, the birth of new interface elements is used to handle the case of moving interfaces.

3. FINITE ELEMENT ANALYSIS OF TIME-DEPENDENT EVENTS

The dynamic equation for a discrete finite element system can be expressed as [12]

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{R}, \tag{10}$$

where $\mathbf{U}$, $\dot{\mathbf{U}}$, and $\ddot{\mathbf{U}}$ are the displacement, velocity, and acceleration column matrices, respectively. $\mathbf{K}$, $\mathbf{C}$, and $\mathbf{M}$ are the global stiffness, damping, and mass matrices, respectively. $\mathbf{K}$ is formed by assembling the local stiffness matrix of each element such as $\mathbf{k}$ in Eq. (7).

The dynamic behavior of objects can be simulated by direct integration methods [5]. Here "integration" means that the above dynamic equation for a discrete finite element system is integrated using a numerical step-by-step procedure. The term "direct" means that the above equation is not transformed to a different form (e.g., modal analysis or first-order system) prior to numerical integration. The main idea of step-by-step integration methods is to loosen the requirement of "satisfaction of the basic dynamic equation at any time t" to the requirement of "satisfaction of the basic dynamic equation at only limited discrete time points: $0, \Delta t, 2\Delta t, \ldots$," where Δt is an incremental step in time chosen to coincide with the incremental step in external loads. In this paper, all kinematic and static variables at different time points are identified by their right subscripts.

The direct integration methods are generally catalogued into explicit and implicit methods. The basic concept of explicit methods is that the solution of $\mathbf{U}_{t+\Delta t}$ is solely based upon the equilibrium equation at time t as follows

$$\mathbf{M}\ddot{\mathbf{U}}_t + \mathbf{C}\dot{\mathbf{U}}_t + \mathbf{K}\mathbf{U}_t = \mathbf{R}_t. \tag{11}$$

The central difference method is a typical explicit method [5]. The major shortcoming of explicit integration methods is that the size of time step Δt is limited by numerical instabilities, which may require the usage of a large number of time steps for some problems.

Implicit methods determine the solution at time $t + \Delta t$ based upon the equilibrium equation at time $t + \Delta t$. These methods provide more accurate results at each time step at the cost of more computer time. Typical implicit integration methods include the Houbolt method [21], the Newmark method [28], and the Wilson-θ method [41]. One weakness of the Houbolt method is that it requires a special starting procedure which is not convenient to users. Both the Newmark method and the Wilson-θ method are extensions to the linear acceleration method, in which the acceleration is assumed to vary linearly from time t to time $t + \Delta t$. These two methods are unconditionally stable for linear cases if $\gamma \geq 0.5; \beta \geq 0.25(0.5 + \gamma)^2$ for the former and $\theta \geq 1.37$ for the latter, where γ and β are parameters of the Newmark method and θ is a parameter of the Wilson-θ method. In this paper, the Newmark scheme is used.

In the dynamic contact-impact analysis, the Lagrange multiplier method and the penalty method have been the two most commonly used approaches [9, 35, 36, 10]. Belytschko and Neal [36] show that the penalty method usually causes less than 25% decrease in time step for stability analysis. In this paper, the penalty approach is used.

4. SIMULATION OF FRACTURE IN OBJECTS

Fracture is a hard problem in simulating the interaction between deformable objects. This problem has been investigated extensively in the area of solid mechanics. However, relatively little work has been done in the area of computer graphics. Terzopoulos *et al.* [37] address this problem to a limited extent. By considering the different goals of solid mechanics and computer graphics, the fracture mechanics approach is not adopted in this paper. Instead, a simple scheme is designed to approximate the fracture propagation for the sole purpose of interactive or real-time visualization.

Generally speaking, two steps are required for the simulation of fracture in objects. The first step is to identify where and when a fracture happens inside an object. This task can be accomplished by using failure criteria of material. The second step is to handle the propagation of fracture after an initial fracture has started.

4.1. Failure Criteria of Material

There are two main types of failure criteria for various types of materials: tensile and shear [12, 13, 19]. For the sake of simplicity, only tensile failure is considered in this paper. One of the most common tensile failure criteria is expressed by the equation

$$\sigma_1 \geq F_s, \tag{12}$$

where σ_1 is the major principal stress inside an object with a specific type of material and F_s is the tensile strength of the material.

4.2. Fracture Propagation

The basic idea is to simulate a fracture propagation as a discrete time-step process. In each time step, the fracture orientation is evaluated by assessing the directions of principal

stresses inside each element. If the failure criterion is met in one element, then the fracture is supposed to happen in the direction of major principal stress in the case of tensile failure.

Let the stress components at a point inside an object be expressed by the matrix

$$\mathbf{T} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}, \tag{13}$$

where $\mathbf{T}$ is the stress tensor whose nine stress components describe the state of stress at a point in an object. According to the material of mechanics [6], the principal stresses satisfy the equations

$$\begin{aligned} l(\sigma_{xx} - \sigma) + m\sigma_{xy} + n\sigma_{xz} &= 0, \\ l\sigma_{yx} + m(\sigma_{yy} - \sigma) + n\sigma_{yz} &= 0, \\ l\sigma_{zx} + m\sigma_{zy} + n(\sigma_{zz} - -\sigma) &= 0, \end{aligned} \tag{14}$$

and

$$l^2 + m^2 + n^2 = 1, \tag{15}$$

where l, m, and n are the direction cosines of principal stress σ with respect to directions X, Y, and Z, respectively.

Let $(\sigma_1, \sigma_2, \sigma_3)$ denote three principal stresses at a point in the object. By convention, $\sigma_1 \geq \sigma_2 \geq \sigma_3$. Since the tensile failure criterion of major principal stress is used in this approach, only the direction cosines of σ_1 is of interest. Substitution of σ_1 into Eq. (14) leads to

$$\begin{aligned} l_1(\sigma_{xx} - \sigma_1) + m_1\sigma_{xy} + n_1\sigma_{xz} &= 0, \\ l_1\sigma_{yx} + m_1(\sigma_{yy} - \sigma_1) + n_1\sigma_{yz} &= 0, \quad \text{and} \\ l_1\sigma_{zx} + m_1\sigma_{zy} + n_1(\sigma_{zz} - -\sigma_1) &= 0, \end{aligned} \tag{16}$$

where (l_1, m_1, n_1) are the direction cosines of σ_1. For the sake of simplicity in the following discussion, Eq. (16) is rewritten as

$$\begin{aligned} a_1 l_1 + b_1 m_1 + c_1 n_1 &= 0, \\ a_2 l_1 + b_2 m_1 + c_2 n_1 &= 0, \quad \text{and} \\ a_3 l_1 + b_3 m_1 + c_3 n_1 &= 0. \end{aligned} \tag{17}$$

Only two equations in Eq. (17) are independent. To solve for (l_1, m_1, n_1), the third independent equation comes from the law of direction cosines

$$l_1^2 + m_1^2 + n_1^2 = 1. \tag{18}$$

If a_1 is not equal to zero, then the first equation in Eq. (17) can be transformed to

$$l_1 = \frac{1}{a_1}(b_1 m_1 + c_1 n_1). \tag{19}$$

Substitution of the above equation into the second equation in Eq. (17) yields

$$\left(\frac{a_2}{a_1}b_1 + b_2\right)m_1 = \left(-c_2 - \frac{a_2}{a_1}c_1\right)n_1. \tag{20}$$

Let $t_1 = (\frac{a_2}{a_1}b_1 + b_2)$ and $t_2 = (-c_2 - \frac{a_2}{a_1}c_1)$. If $t_1 \neq 0$, then

$$m_1 = \frac{t_2}{t_1}n_1. \tag{21}$$

Substitution of the above equation back to Eq. (19) leads to

$$l_1 = \frac{1}{a_1}\left(b_1\frac{t_2}{t_1} + c_1\right)n_1. \tag{22}$$

Let $t_3 = \frac{1}{a_1}(b_1\frac{t_2}{t_1} + c_1)n_1$. By the substitution of Eq. (22) and (20) into Eq. (18), one gets

$$n_1^2 = \frac{1}{t_3^2 + \left(\frac{t_2}{t_1}\right)^2 + 1}. \tag{23}$$

After n_1^2 is obtained, it is easy to determine the values of m_1^2 and l_1^2 by using Eqs. (21) and (18). Theoretically, the orientation of fracture surface in an element could be in any direction designated by l_1, m_1, and n_1. To save the computation cost, only three directions in X, Y, and Z are considered as an initial representation of fracture propagation.

To determine the direction of the fracture surface, it is required to determine which of the directions X, Y, and Z is closest to the direction of σ_1 by identifying the smallest magnitude among l_1^2, m_1^2, and n_1^2. For instance, if n_1^2 is of the smallest magnitude among the three, then the normal of the fracture surface is considered to be approximately along the Z direction.

The above results are obtained under the condition of $t_1 \neq 0$. If $t_1 = 0$ and $t_2 \neq 0$, then

$$\begin{aligned} n_1 &= 0, \\ l_1 &= \frac{b_1}{a_1}m_1, \quad \text{and} \\ l_1^2 + m_1^2 &= 1. \end{aligned} \tag{24}$$

In the cases where $t_1 = 0$ and $t_2 = 0$ or where $a_1 = 0$, the above procedure is repeated to sequentially try a_2 in the second equation and a_3 in the third equation of Eq. (17). If $a_1 = a_2 = a_3 = 0$, there is no unique solution for l_1^2, m_1^2, and n_1^2, and in this paper one solution is randomly picked from ($l_1^2 = 1, m_1^2 = 0, n_1^2 = 0$), ($l_1^2 = 0, m_1^2 = 1, n_1^2 = 0$), and ($l_1^2 = 0, m_1^2 = 0, n_1^2 = 1$).

Once the σ_1 in an element reaches the tensile strength of the material, the failure of the element happens. After identifying the direction to which the σ_1 is closest, the element is split into two elements, as shown in Fig. 11. In this figure, the geometric configuration at the upper position refers to the element before a tensile failure. In the presence of a failure, the

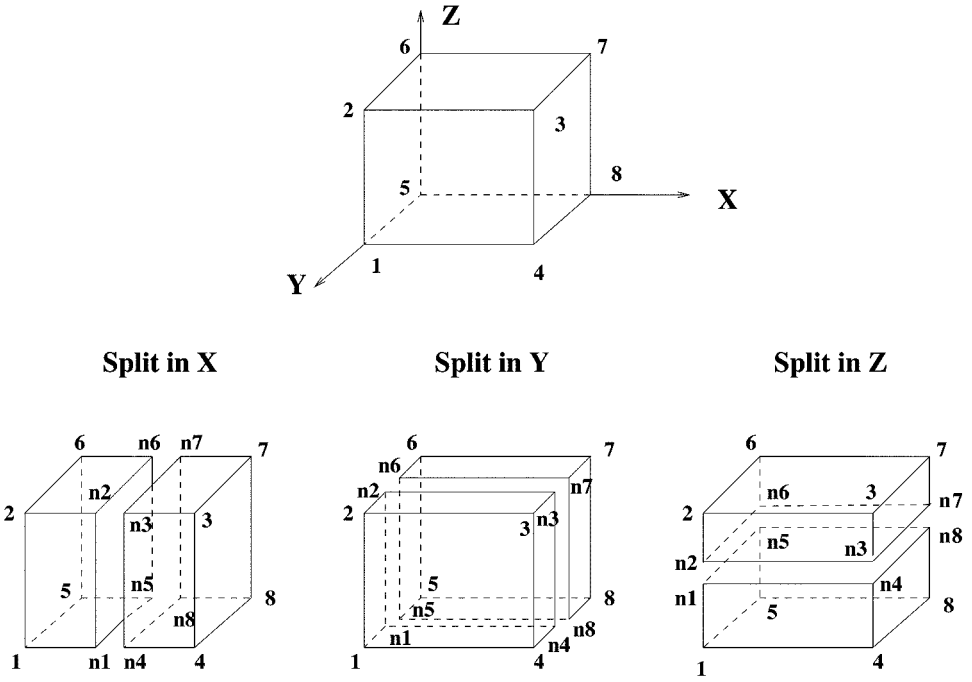

FIG. 11. Split of an element with tensile failure.

element might be split in one of X, Y, or Z directions. This is illustrated by the three sub-diagrams at the lower part of Fig. 11. The element originally has eight-cornered nodes. Eight new nodes $n1, n2, \ldots, n8$ are generated in a split process such that one original element is divided into two elements. These newly generated nodes are not connected to other nodes except those within the element. In this way, the geometric coherence is automatically guaranteed.

The split process can take place in one of three coordinate directions depending upon which direction σ_1 is closest to. As the analysis goes through different time steps, the accumulation of cracks inside an object can be simulated. The term "Element-Split Scheme" is used to designate the above idea which can be extended to a finer split in any direction with the cost of more information keeping.

5. EXPERIMENTAL SIMULATIONS AND EVALUATION

A basic animation, FEANIM 1.0 (Finite Element ANIMation), system has been developed to simulate the dynamic behavior and interaction between deformable objects. It is written in the C language. The implemented code runs on a DEC Alpha Server 2100 with four 200 MHz CPUs and 256 MB of RAM under the Digital UNIX operating system. The main procedures of using FEANIM 1.0 are stated as follows. First, a data script needs to be created, which includes the information of nodes, elements, material properties, boundary conditions, external forces, specific displacements, initial speeds and accelerations, number of incremental steps, etc. Next, FEANIM is run by feeding it a suitable data script. The

output of FEANIM is a rendering script for the input of RayShade, a rendering package available at "ftp://graphic.stanford.edu/pub/rayshade/." The output of RayShade is one or more frames of image files in RLE (Run Length Encoding). Then, MPEG_encode is used to concatenate these RLE files into an mpeg animation file. Finally, MPEG_play is used to view the animation sequence of the mpeg file.

In this paper, the total CPU time refers to user time + system time of FEANIM excluding the time in rendering, because different surface properties of the same shape of an object can make the rendering time varying considerably.

5.1. Test Case 1: Impact of an Elastic Cube on a Deformable Ground

An elastic cube is first raised to a certain height above a deformable ground, and is then dropped to hit the ground, as shown in Fig. 12. The local stiffness matrix **k** for the elements

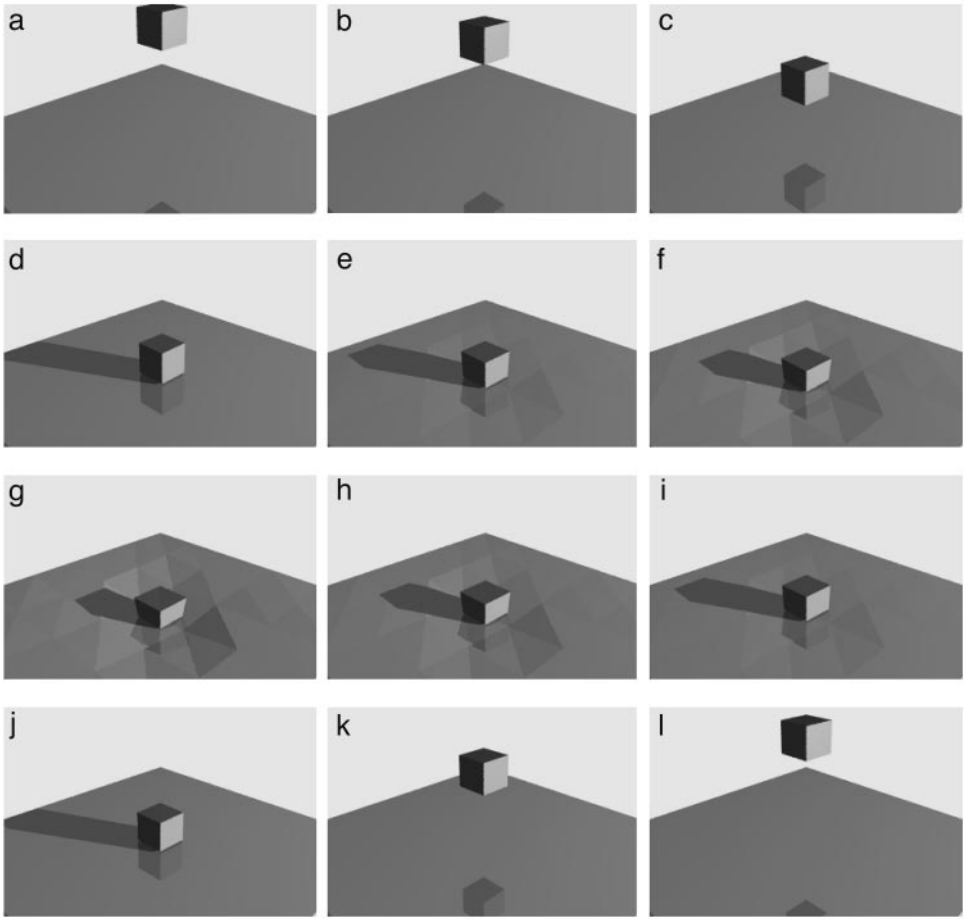

FIG. 12. Collision response between an elastic cube and a deformable ground. 11 steps. Execution of FEANIM: user time = 68 s, system time = 103 s. (a) step 0; (b) step 1; (c) step 2; (d) step 3; (e) step 4; (f) step 5; (g) step 6; (h) step 7; (i) step 8; (j) step 9; (k) step 10; (l) step 11.

of the cube and the ground is chosen as the following elastic constitutive matrix

$$\mathbf{k} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}, \tag{25}$$

where E is the Young's modulus (the elasticity modulus) and ν denotes Poisson's ratio. E and ν of the cube are 10,000 KN/m^2 and 0.3, respectively. E and ν of the ground are 5000 KN/m^2 and 0.3, respectively. The densities of the cube and the ground are both set to be 300 Kg/m^3. When the cube collides with the ground, the whole body of the cube is compressed in the vertical direction and its size is expanded in the horizontal plane because of its inertial force. After the vertical velocity of the cube reduces to zero, the cube bounces up as an elastic object. The ground is fixed at its four boundary lines, only two of which can be seen in Fig. 12. The rest of the ground is allowed to deform. When the cube hits the ground, the ground deforms most at the area hit by the cube. The deformation of the ground gradually decreases from the hit area to the boundary lines where the deformation becomes zero. Figures 12d through 12j show this pattern of deformation.

Figure 13 shows the deformation of the mesh of the ground. Point-to-surface elements are represented by shaded triangular elements. Each pair of these triangular elements form an approximate four-sided element. The generation and destruction of point-to-surface elements are clearly shown in the figure, which implies the correctness of the framework for interface to a certain extent.

5.2. Test Case 2: Bouncing up a Deformable Ball from a Deformable Ground

A deformable ball is dropped from a certain height above a deformable ground, as shown in Fig. 14. The Young's modulus and Poisson's ratio of the cube are 40,000 KN/m^2 and 0.3, respectively. The Young's modulus and Poisson's ratio of the ground are 15,000 KN/m^2 and 0.3, respectively. The densities of the cube and the ground are the same as those in test case 1. The contact area is a point as soon as the ball hits the ground and then expands to a circle, as illustrated by Fig. 14d through 14g. The deformation of the ball is not uniform throughout its body. Its lower portion experiences larger deformation, while the deformation of the upper portion is relatively small. This deformation pattern is reasonable for an elastic ball.

The boundary condition of the ground is the same as that in Case 1. When the ball hits the ground, both the ball and ground deform in the vertical direction. At the moment the vertical velocity of the ball decreases to zero, the deformation of both reaches the maximum, as shown in Fig. 14g. As the ball leaves the ground, its deformation restores to zero.

5.3. Test Case 3: A Ball Colliding a Hard Ground Obliquely

When a ball hits a hard ground obliquely, it should also bounce up obliquely with respect to the ground. For the sake of clarity, a two-dimensional view is provided in Fig. 15.

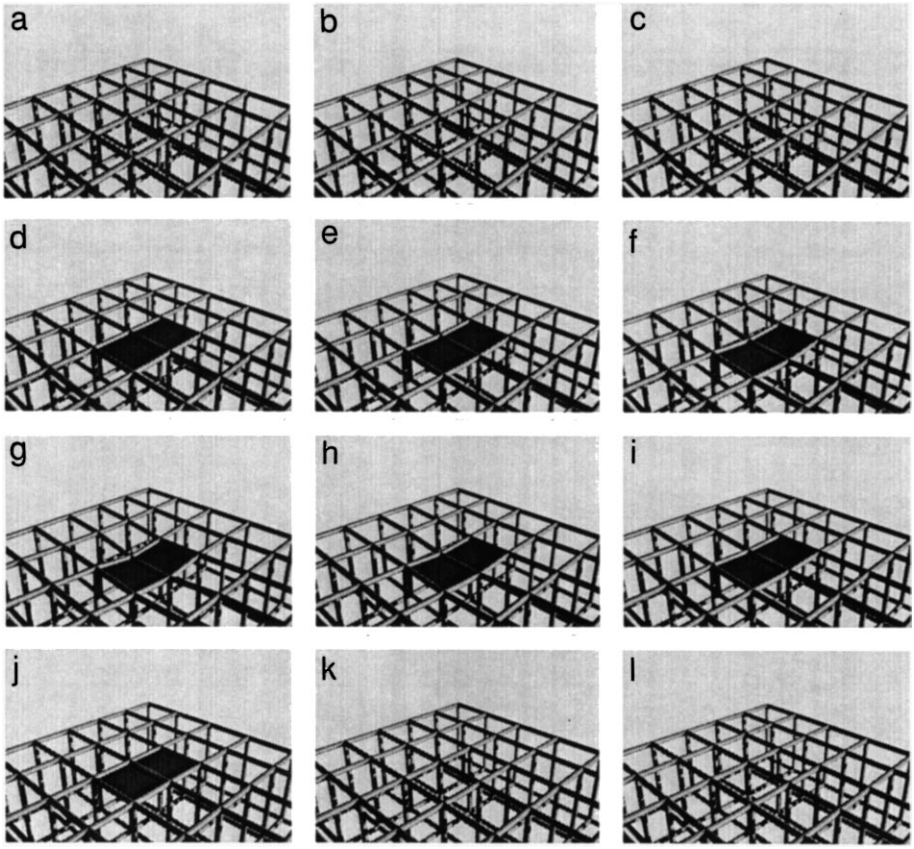

FIG. 13. Collision response between an elastic cube and a deformable ground. The wire-frame shows the mesh of the ground. Each square element consists of two point-to-surface elements between the cube and the ground. (a) step 0; (b) step 1; (c) step 2; (d) step 3; (e) step 4; (f) step 5; (g) step 6; (h) step 7; (i) step 8; (j) step 9; (k) step 10; (l) step 11.

The original and final positions of the ball are shown in Fig. 15a and 15k, respectively. It has an initial horizontal speed toward the left and an initial acceleration downward to the ground. After the ball hits the ground, it should bounce upward and leftward, because of the influence from the initial leftward speed. This tendency is indicated by Fig. 15c through 15f.

5.4. Test Case 4: Fracture of a Cantilever Beam

The evaluation of fracture propagation is generally difficult because no analytical solution is available to describe it. A partial evaluation is to use the formula in mechanics of material for the estimation of the place where the fracture of an object likely happens first. Such an evaluation can be accomplished only in cases where the analytical solution of stresses inside the object is available, because various types of failure criteria can be evaluated on the basis of stress information.

Figure 16 shows the fracture propagation in the cantilever beam. The Young's modulus, Poisson's ratio, and density of the beam are 20,000 KN/m^2, 0.3, and 300 Kg/m^3, respectively. The beam is fixed at its left end and an incremental vertical load is applied downward at

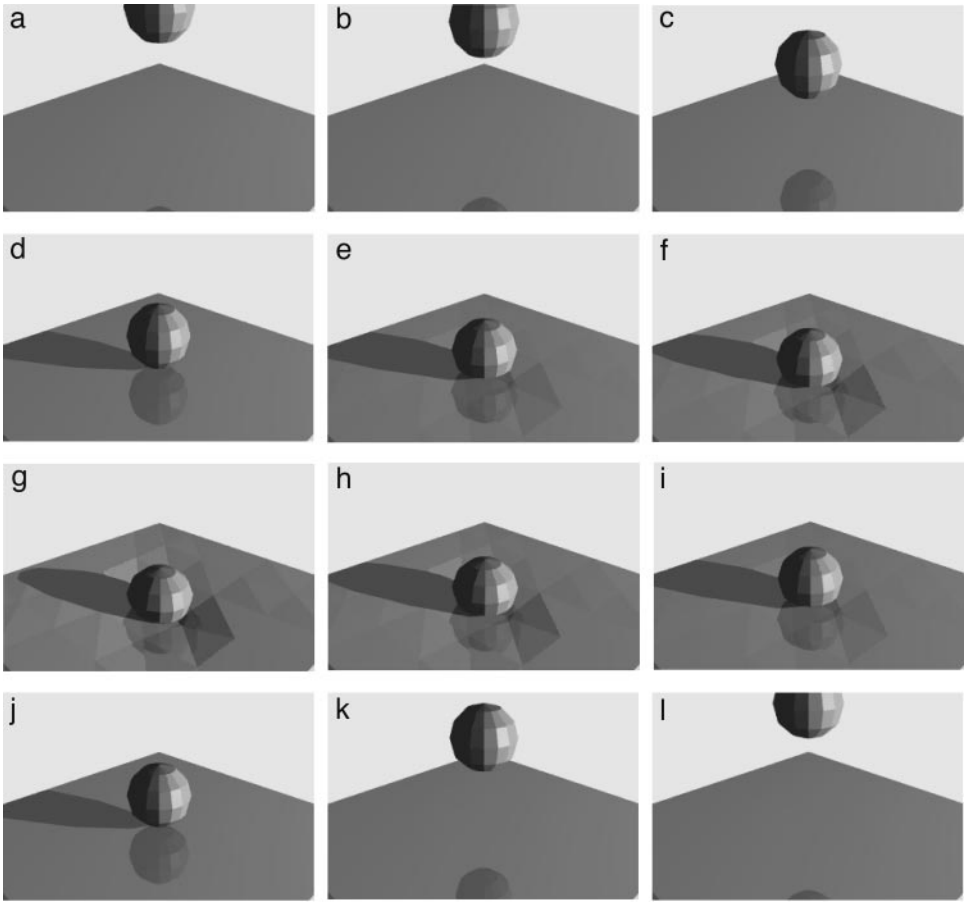

FIG. 14. Collision response between an elastic sphere and a deformable ground. 11 steps. Execution of FEANIM: user time = 9274 s, system time = 19159 s. (a) step 0; (b) step 1; (c) step 2; (d) step 3; (e) step 4; (f) step 5; (g) step 6; (h) step 7; (i) step 8; (j) step 9; (k) step 10; (l) step 11.

its right end. Since there is a U-shaped cut at the middle of the beam, stress concentrations likely exist in the area around the cut. Furthermore, the stress concentrations at the left corner of the cut should be higher than those at the right corner, because the larger bending moment exists at the left corner. Figure 17 shows the size of the beam. According to Eq. (57) in [39], the normal stress in the cross section of the beam is calculated by

$$\sigma_x = \frac{My}{I_z}, \tag{26}$$

where σ_x is the normal stress in the cross section. M is the bending moment and I_z is the moment of inertia with respect to the neutral axis Z. For the rectangular beam, I_z is given by

$$I_z = \frac{bh^3}{12}, \tag{27}$$

where b and h are the width and height of the beam, respectively. If no stress concentration is considered for the sake of simplicity, the maximum normal stress in cross section 1 in

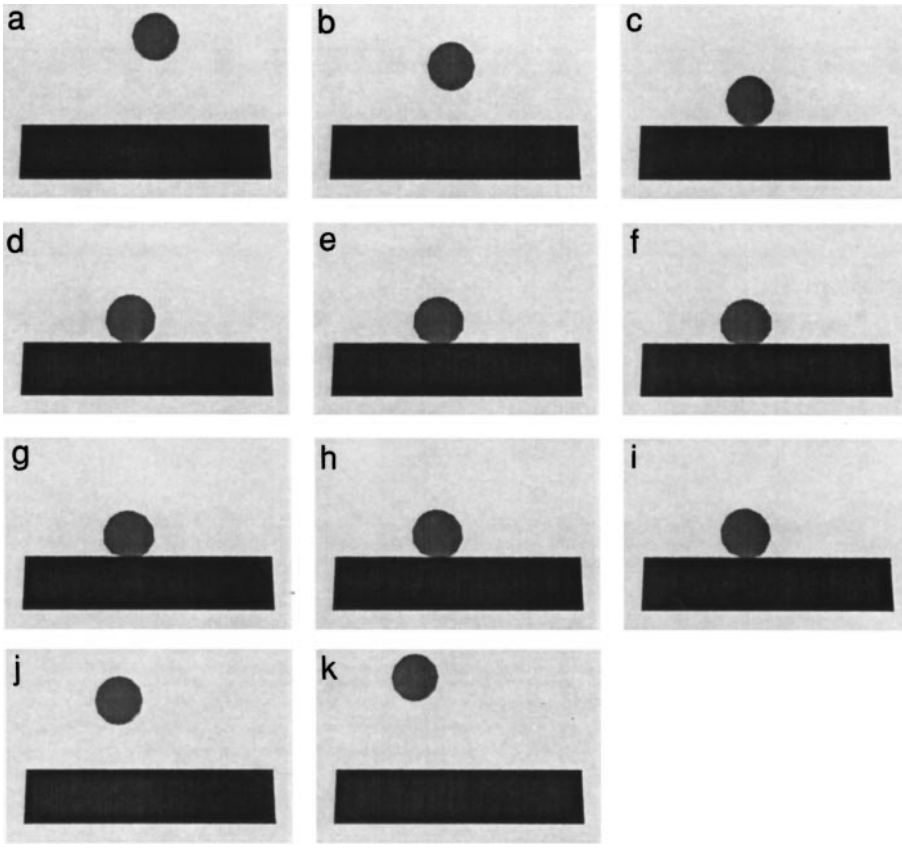

FIG. 15. Collision response between an elastic ball and a hard ground in a two-dimensional view. 10 steps. Execution of FEANIM: user time = 17 s, system time = 31 s. (a) step 0; (b) step 1; (c) step 2; (d) step 3; (e) step 4; (f) step 5; (g) step 6; (h) step 7; (i) step 8; (j) step 9; (k) step 10.

Fig. 17 is

$$(\sigma_x)_{\max} = \frac{My_{\max}}{I_z} = \frac{(3P)(0.5)}{\frac{1\cdot 1^3}{12}} = 18P. \tag{28}$$

As to cross section 2, the maximum normal stress is

$$(\sigma_x)_{\max} = \frac{My_{\max}}{I_z} = \frac{(1.75P)(0.33)}{\frac{1\cdot 0.66^3}{12}} = 24.1P. \tag{29}$$

Since the maximum normal stress in cross section 2 is larger than that in cross section 1, fracture should happen first in cross section 2. This is supported by Figs. 16a through 16f. Therefore, it is reasonable for cracks to be generated first around the left corner. Next the cracks expand at the left and in the meantime some cracks also occur in the root area of the beam.

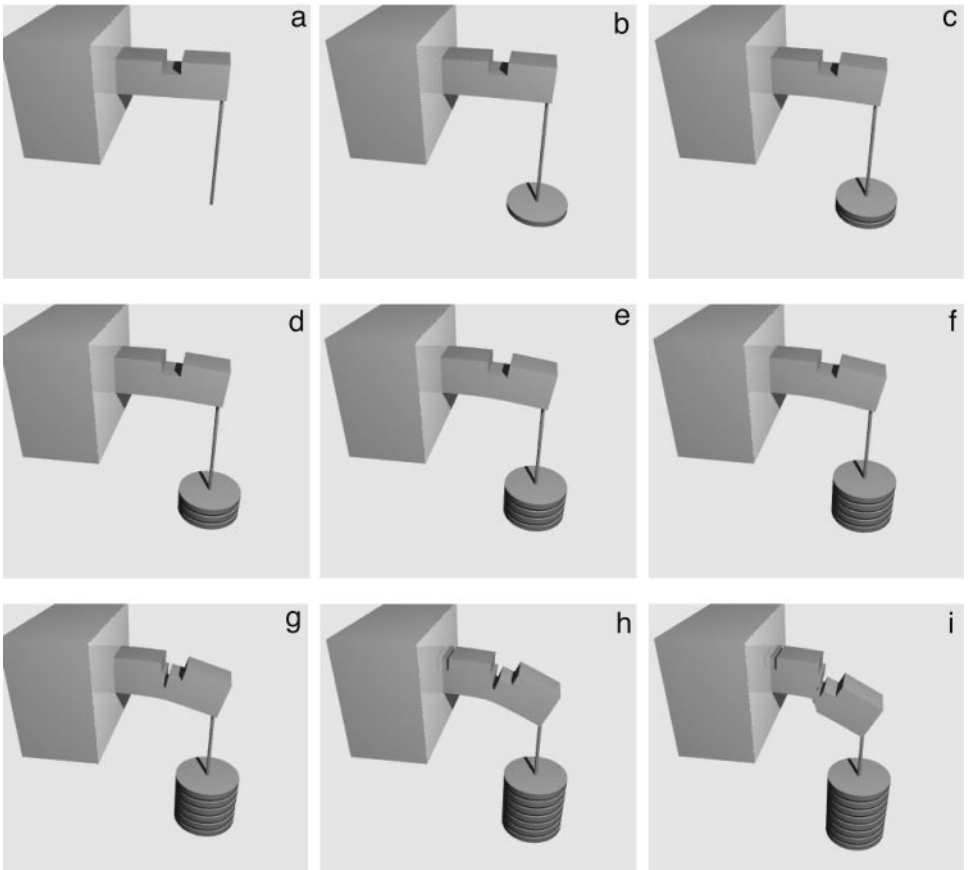

FIG. 16. Fracture propagation in a bent beam. 8 steps. Execution of FEANIM: user time = 253 s, system time = 97 s. (a) step 0; (b) step 1; (c) step 2; (d) step 3; (e) step 4; (f) step 5; (g) step 6; (h) step 7; (i) step 8.

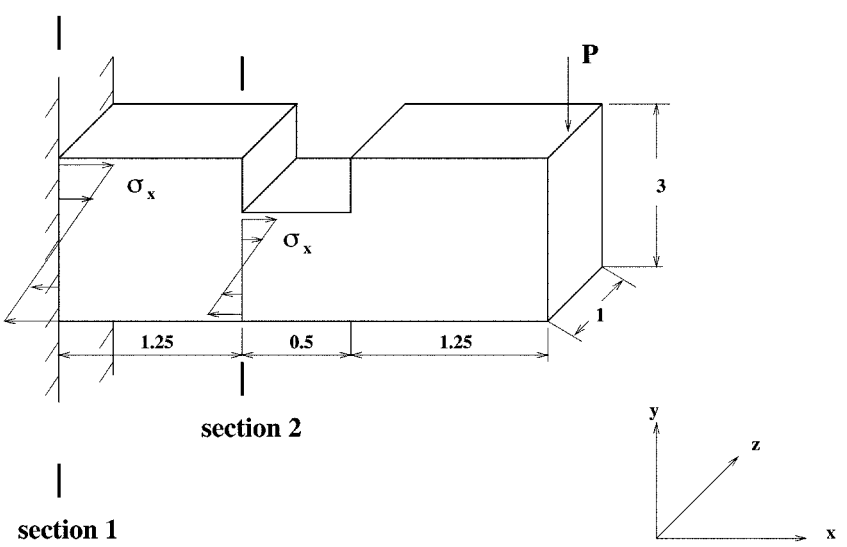

FIG. 17. Geometric size of the beam.

6. CONCLUSIONS

The main contributions of this paper include a new framework of interface elements and a simple approach to simulating fracture propagation.

The new framework of interface elements provides a flexible way to simulate various types of interfaces (point-to-point, point-to-surface, and surface-to-surface) in the domain of finite element analysis. It uses two types of elements, point-to-point and point-to-surface. The point-to-point element is a two-node rod-like element allocated statically prior to an analysis, while the point-to-surface is a four-node tetrahedron element whose birth and death are dynamically controlled by a specific mechanism during the numerical analysis. Different friction and tension failure criteria can be easily incorporated into the framework. It avoids the shortcoming associated with the *spring approach* when the surface nodes on the two colliding objects are not contiguous to each other. It also avoids the requirement for small time step with the *pinball approach*.

A simple "Element-Split Scheme" is proposed to give a conceptual representation of the fracture propagation inside objects. By the constraint of time-saving requirements, the split in only three directions is considered in the animation sequence. But the concept is easily expanded to any finite number of directions for a finer simulation. The simulated fracture process of the cantilever beam in test case 4 is supported by an approximate analysis on the basis of mechanics of material.

The major computation cost of the approach comes from solving the system dynamic equation, i.e., Eq. (10). Using the skyline profile scheme is one way to implement the solution. A more efficient implementation is to adopt a sparse matrix solver which can avoid the computation of zero entries both outside and inside the skyline profile of a global stiffness matrix.

REFERENCES

1. W. W. Armstrong and M. Green, The dynamics of articulated rigid bodies for purposes of animation, *Visual Computer* **1** (1985), 231–240.
2. D. Baraff, Analytical methods for dynamic simulation of non-penetrating rigid bodies, *Computer Graphics (Proc. SIGGRAPH)* **23**, No. 3 (1989), 223–232.
3. D. Baraff and A. Witkin, Dynamic simulation of non-penetrating flexible bodies, *Computer Graphics (Proc. SIGGRAPH)* **26**, No. 2 (1992), 303–308.
4. R. Barzel and A. Barr, Modeling with dynamic constraints, "ACM SIGGRAPH'87 Course Notes," Anaheim, CA, Vol. 17, 1987.
5. K. J. Bathe and E. L. Wilson, "Numeric Methods in Finite Element Analysis." Prentice-Hall, New York, 1976.
6. A. P. Boresi, R. J. Schmidt, and O. M. Sidebottom, "Advanced Mechanics of Materials." Wiley, New York, 1993.
7. M. Cani-Gascuel and M. Desbrun, Animation of deformable models using implicit surfaces, *IEEE Trans. Visualization Computer Graphics* **3**, No. 1 (1997), 39–50.
8. M. Carigan, Y. Yang, N. M. Thalmann, and D. Thalmann, Dressing animated synthetic actors with complex deformable clothes, *Computer Graphics (Proc. SIGGRAPH)* **26**, No. 2 (1992), 99–104.
9. N. J. Carpenter, R. L. Taylor, and M. G. Katona, Lagrange constraints for transient finite element surface contact, *Int. J. Numerical Methods Eng.* **32** (1991), 103–128.
10. A. B. Chaudhary and K. J. Bathe, A solution method for static and dynamic analysis of three-dimensional contact problems with friction, *Comput. Struct.* **24** (1986), 865–873.
11. J. Cohen, M. Lin, D. Manocha, and M. Ponamgi, I-collide: An interactive and exact collision detection system for large-scale environment, *in* "Proceedings of ACM Interactive 3D Graphics Conference," pp. 189–196, 1995.

12. C. S. Desai and J. T. Christian, "Numerical Methods in Geotechnical Engineering," McGraw–Hill, New York, 1977.
13. D. C. Drucker and W. Prager, Soil mechanics and plastic analysis or limit design, *Q. Appl. Math.* **10**, No. 2 (1952), 157–165.
14. P. Faloutsos, M. V. D. Panne, and D. Terzopoulos, Dynamic free-form deformations for animation synthesis, *IEEE Trans. Visualization Computer Graphics* **3**, No. 3 (1997), 201–213.
15. A. Fournier and W. T. Reeves, A simple model of ocean waves, *Computer Graphics (Proc. SIGGRAPH)* **20**, No. 4 (1986), 75–84.
16. R. S. Gallagher and J. Nagtegaal, An efficient 3d visualization technique for finite element models and other coarse volumes, *Computer Graphics (Proc. SIGGRAPH)* **23**, No. 3 (1989), 185–194.
17. M. Girard and A. A. Maciejewski, Computational modelling for the computer animation of legged figures, *Computer Graphics* **9**, No. 3 (1985), 263–270.
18. J. P. Gourret, N. M. Magnenat-Thalmann, and D. Thalmann, Simulation of object and human skin deformation in a grasping task, *Computer Graphics (Proc. SIGGRAPH)* **23**, No. 3 (1989), 21–30.
19. R. Hill, "The Mathematical Theory of Plasticity," Clarendon Press, Oxford, 1950.
20. C. M. Hoffmann and J. E. Hopcroft, Simulation of physical systems from geometric models, *IEEE J. Robotics Automation* **RA-3**, No. 3 (1987), 194–206.
21. J. C. Houbolt, A recurrence matrix solution for the dynamic response of elastic aircraft, *J. Aeronautical Sci.* **17** (1950), 540–550.
22. P. M. Issacs and M. F. Cohen, Controlling dynamic simulation with kinematic constraints, behaviour functions, and inverse dynamics, *Computer Graphics* **21**, No. 4 (1987), 215–224.
23. T. Kurihara, K. Anjyo, and D. Thalmann, Hair animation with collision detection, *in* "Models and Techniques in Computer Animation," pp. 128–138, 1993.
24. B. Lafleur, M. N. Thalmann, and D. Thalmann, Cloth animation with self-collision detection, *in* "Modeling in Computer Graphics" (T. L. Kunii, Ed.), pp. 179–185, 1992.
25. C. Lanczos, "The Variational Principles of Mechanics." University of Toronto Press, Toronto, 1949.
26. L. Ling, M. Damodaran, and R. K. L. Gay, A quasi-steady force model for animating cloth, *in* "Models and Techniques in Computer Animation," pp. 128–138, 1993.
27. M. Moore and J. Wilhelms, Collision detection and response for computer animation, *Computer Graphics (Proc. SIGGRAPH)* **22**, No. 4 (1987), 279–288.
28. N. M. Newmark, A method of computation for structural dynamics, *A.S.C.E., J. Eng. Mech. Div.* **85** (1959), 67–94.
29. A. Pentland and J. Williams, Good vibrations: Modal dynamics for graphics and animation, *ACM Trans. Computer Graphics* **23**, No. 3 (1989), 215–222.
30. J. C. Platt and A. H. Barr, Constraint methods for flexible models, *Computer Graphics (Proc. SIGGRAPH)* **22**, No. 41 (1988), 279–288.
31. W. T. Reeves, Particle systems-a technique for modeling a class of fuzzy objects, *ACM Trans. Graphics* **2**, No. 2 (1983), 91–108.
32. W. T. Reeves and R. Blau, Approximate and probabilistic algorithms for shading and rendering structured particle systems, *Computer Graphics (Proc. SIGGRAPH)* **19**, No. 3 (1985), 313–322.
33. K. Sim, Particle animation and rendering using data parallel computation, *Computer Graphics (Proc. SIGGRAPH)* **24**, No. 4 (1986), 405–413.
34. J. C. Simo, P. Wriggers, and R. L. Taylor, A perturbed Lagrangian formulation for the finite element solution of contact problems, *Comp. Methods Appl. Mech. Eng.* **50** (1985), 163–180.
35. T. Belytschko and J. I. Lin, A three-dimensional impact-penetration algorithm with erosion, *Comput. Struct.* **25** (1987), 95–104.
36. T. Belytschko and M. O. Neal, Contact-impact by pinball algorithm with penalty and Lagrangian methods, *Int. J. Numerical Methods Eng.* **31** (1991), 547–572.
37. D. Terzopoulos and K. Fleischer, Deformable models, *Visual Computer* **4**, No. 6 (1988), 306–331.
38. D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, Elastically deformable models, *Computer Graphics (Proc. SIGGRAPH)* **21**, No. 4 (1987), 205–214.

39. S. P. Timoshenko, “Strength of Materials,” D. Van Nostrand, Princeton, NJ, 1955.
40. J. Wilhelms and B. A. Barsky, Using dynamic analysis to animate articulated bodies such as humans and robots, *in* “Proc. Graphics Interface ’85,” pp. 97–104, Montreal, Canada, 1985.
41. E. L. Wilson, I. Farhoomand, and K. J. Bathe, Nonlinear dynamic analysis of complex structures, *Int. J. Earthquake Eng. Structural Dynamics* **1** (1973), 241–252.
42. A. Witkin and W. Welch, Fast animation and control for non-rigid structures, *Computer Graphics (Proc. SIGGRAPH)* **24**, No. 4 (1990), 243–252.
43. Y. Yang and M. N. Thalmann, An improved algorithm for collision detection in cloth animation with haman body, *in* “Proc. Pacific Graphics ’93,” pp. 237–251, 1993.