Robotica (2004) volume 22, pp. 223–230. © 2004 Cambridge University Press
DOI: 10.1017/S0263574703005617 Printed in the United Kingdom

On interacting with physics-based models of graphical objects

Shahram Payandeh, John Dill and Zhu Liang Cai

Experimental Robotics and Graphics Laboratory, School of Engineering Science, Simon Fraser University, 8888 University Drive, Burnaby, British Columbia (CANADA), V5A 1S6. E-mail: aharam@cs.sfu.ca, dill@cs.sfu.ca

(Received in Final Form: September 12, 2003)

SUMMARY

Enhancing graphical objects whose behaviors are governed by the laws of physics is an important requirement in modeling virtual physical environments. In such environments, the user can interact with graphical objects and is able to either feel the simulated reaction forces through a physical computer interface such as a force feedback mouse or through such interactions, objects behave in a natural way. One of the key requirements for such interaction is determination of the type of contact between the user controlled object and the objects representing the environment. This paper presents an approach for reconstructing the contact configuration between two objects. This is accomplished through usage of the time history of the motion of the approaching objects for inverse trajectory mapping of polygonal representation. In the case of deformable objects and through usage of mass-spring-damper system this paper also presents a special global filter that can map the local deformation of an object to the adjacent vertices of polygonal mesh. In addition to offering a fast computational framework, the proposed method also offers more realistic representation of the deformation. The results of this paper are shown through detailed examples and comparison analysis using different computational platforms.

KEYWORDS: Contact reconstruction; Physics based modeling; Deformable objects; Mass-spring-damper system; Global filter; Realistic deformation.

1. INTRODUCTION

Simulating and interacting with objects possessing physical–dynamical–attributes in addition to geometrical attributes is an important area of research in modeling *virtual reality*. Applications include robotics, path planning, animation, computer games and training systems. It is particularly important in training systems for the user/trainee to be able to feel the forces created during interaction with a system, in addition to experiencing the usual visual feedback and interaction. Such feedback can be provided with a special-purpose mechanical user interface, commonly referred to as a *haptic interface*. Interaction with such a system consists of integration and synchronization of several components. Two important aspects of such interactions are the notion of contact/impact event reconstruction and augmentation of the graphical object (visual rendering) with physical attributes (haptic rendering).

Detection of contact between geometric objects has been investigated extensively in the literature. The basic methodology for detecting collisions between objects is based on utilizing a polygonal mesh representation of their surfaces. For example, Minkowski difference and convex optimization are used to compute the distance between convex polyhedra by finding the closest points.[1–3] More recent work is based on tighter-fitting bounding volumes. For example, references [4, 5] has developed an approach for interference detection based on oriented bounding boxes, which more closely approximate object geometry. This approach was later extended to deal with the case where objects can have multiple contacts with each other. Other approaches for detecting contact between objects are based on Constructive Solid Geometry (CSG). For example, references [6, 7] introduced an efficient approach based on CSG for determining whether the intersection is empty by utilizing space partitioning and bounding boxes. Still other approaches based on the voxel-based objects have been proposed. One of the key benefits of voxel-based approaches is that they are conceptually simple to implement. For example, collisions are detected automatically when a voxel address from one object tries to write into an occupancy map cell that is already occupied by the voxel address of another object.[8] There are other methods which address the speed and robustness of the algorithm for example presented in references [9] and [10]. For further reference, a very good survey of collision detection can be found in reference [11].

The next step in utilization of the contact detection is the reconstruction of the contact configuration between the possible objects, which is one of the themes of this paper. Here, the basic outputs from any collision detection algorithm are the probability of the possible polygons of the two objects that might be in the interference mode. In this paper, an approach for reconstructing contact configuration is presented. The approach is based utilization the inverse trajectory mapping of the two contacting objects and usage of the data structure associated with the interfering polygons, i.e. half-edge data structure.

In addition to representing an object's geometrical and appearance attributes, it is important to be able to represent their physical attributes such as mechanical, thermal and fluid properties, which can in general be modeled using the notion of continuum mechanics.[12] However, the practical implementation of such models requires discretization of such continuum models. The most general approach for representing such discretization is based on the notion of the finite element[13–16] or boundary element method.[17] In general, this approach for modeling objects produces very good results but the high computational cost can hinder its practical application. Here, each element of the body is represented by the constitutive equations describing the physical phenomenon associated with the behavior of the object. There have been some efforts to reduce the real-time requirements of the modeling approaches using finite-element method, for example see reference [18]. The other main draw-back of modeling using the finite element approach is that, in general, definition of non-linear strain energy functions need to be used for proper modeling of organs and tissues. These models require material property constants that are very difficult to obtain (either live or otherwise).

Another approach, which can be viewed as a subclass of the general finite element method, is the notion of mass-spring-damper systems.[19,20] Here, similar to the finite element method, the object is represented by a number of discrete elements composed of vertices and edges. Each vertex has a mass element and each edge consists of a spring and damper element. The behavior of the object in response to an external force is the behavior of this mass-spring-damper system. One main reason for its popularity is the simplicity of modeling. Other reasons are the relative ease of parallelizing of the model for improved computational performance. In addition, through proper modeling and fine-tuning the stiffness representation with the finite element representation, it is possible to accomplish realistic behavior of the organs and tissues. Various numerical integration schemes have also been proposed for efficient and stable response of such systems. For example, Miller[21] proposed a model for animating legless figures, such as snakes and worms, using mass-spring systems. Animating spring tensions simulates muscle contractions. Directional friction forces are introduced in the model for resulting locomotion. Provot[22] proposed a model for animating cloth objects using a network of springs and masses. Based on the model for the global mesh of deformable object based on the mass-spring-damper model, this paper presents a novel approach for determining the deformation response of the object which takes into account the global stiffness model of the object and hence filter and distribute the local deformation in resulting in a more realistic behavior. In addition, the data structure of the proposed model allows deformation solution of the object when there are multiple objects interacting with the deformable object.

The paper is organized as follows: Section 2 presents a method for reconstructing the contact configuration between objects having determined the designated interfering polygons using any available collision detection algorithm; Section 3 presents an approach for modeling and solving deformation of an object using the novel global filtering algorithm for more realistic representation of the deformation and Section 4 presents concluding remarks. Each section also presents relevant simulation results.

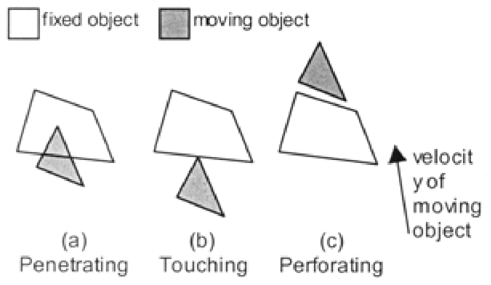

Fig. 1. Possible object configurations after a positive collision query.

2. RECONSTRUCTION OF CONTACT CONFIGURATION

Most contact detection algorithms assume planar polygonal objects (i.e. objects whose surfaces are collections of planar polygons, usually triangles) and handle collision queries by producing an array of indices of colliding triangles.[11] To properly model both the mechanics of interaction between the bodies and the behavior of the objects after deformation, knowledge of the exact contact configuration, i.e. first point, line or surface contacts and pre- and post- velocity trajectories of the moving object is required.

Typically a collision detection routine is queried in a loop to test for collision after each incremental change in position of the moving object. In general, one of three possible results is returned (Figure 1):

(a) The objects inter-penetrate.
(b) The objects' boundaries touch.
(c) The objects perforate (one object passes completely through another object).

Cases (a) and (c) are not what the system expects to obtain. Rather, the system expects the pure touch collision case (b). We must find a way to avoid cases (a) and (c), and to generate an exact collision as in case (b). For case (c), the solution is to reduce the step length to be less than the bounding box's minimum edge length among all objects. We describe a method for case (a).

2.1. *Reconstruction method*

The steps for an algorithm to deal with case (a) are:

- Find whether the objects penetrate or touch.
- Find the vector which repositions the moving object to the point where it first touches, if they penetrate (*reversing vector*).
- Find the first contact points, lines (edges) or polygons (surfaces) as a contact localization sub-algorithm, Figure 2.

Definitions:

$V_{f[i]}$ and $V_{m[i]}$ represent the ith vertex on the fixed and moving objects, respectively;
$E_{m[j]}$ and $E_{f[j]}$ represent the jth line segment on the moving and fixed objects, respectively;
$l_{m[i]}$ and $l_{f[i]}$ represent the ith (jth) line through vertex $V_{m[i]}$ ($V_{f[i]}$), parallel to the object's velocity vector $\boldsymbol{D}$.
P_{ij} is the intersection of line $l_{m[i]}$ and edge $E_{f[j]}$.

First we consider the relationships between vertices $V_{m[i]}$ and line segments $E_{f[i]}$. We define P_{ij} as the intersection point

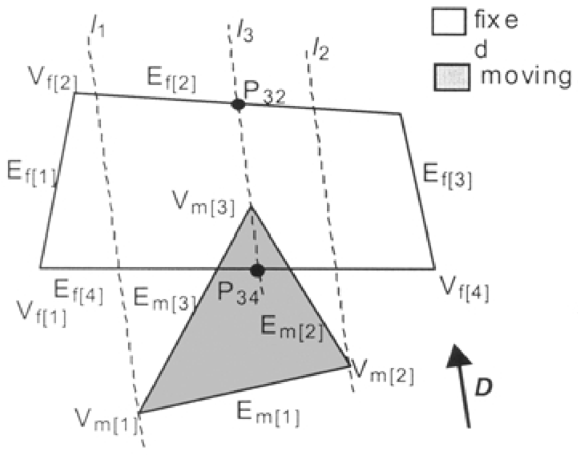

Fig. 2. Collision projection in 2D.

between line $l_{m[i]}$ and the extended line corresponding to line segment $E_{f[j]}$, and test to see if P_{ij} is on the segment $E_{f[j]}$ itself. If not, P_{ij} can be ignored. The penetrating vectors V_{ij} can then be obtained using:

$$V_{ij} = (P_{ij} - V_{m[i]}) \tag{1}$$

Then we find the maximum *penetrating distance* and then the resulting reversing vector. Penetrating distance values d_{ij} are given by:

$$d_{ij} = V_{ij} \cdot (-D_u) \tag{2}$$

where D_u is a unit vector in the direction of D. If $d_{ij} \geq 0$, the corresponding vertex $V_{m[i]}$ penetrates the fixed polyhedron. From the set of d_{ij}'s, a maximum value can be found. Note that at least one vertex will have this maximum value. The set of vertices $V_{m[i]}$ (with this maximum d) is regarded as the set of contact points. To determine the type of contact (vertex, line or polygon touch) we examine *neighboring* contact points, where neighboring means in order as we travel around the polygon boundary. If neighboring contact points do not share an edge or polygon, they are touch vertices; if they share a line segment, an edge touches; and if neighboring contact points form a polygon, a face (polygon) contact is said to occur. Any V_{ij} corresponding to the maximum distance is the reversing vector.

(i) Solution to special situation. The complete contact configuration problem is fully addressed, except for certain singular situations, shown in Figure 3 where the moving object's colliding vertex/vertices is/are not inside the fixed object.

The procedure to handle this case is similar to the above, except instead of considering the fixed object's $\boldsymbol{V}_{m[i]}$, we consider the moving object's $\boldsymbol{V}_{m[i]}$, the line segments are changed from the fixed object's $E_{f[j]}$ to the moving object's $E_{m[j]}$, and the lines corresponding to the moving object $l_{m[i]}$ are also changed to the fixed object's $l_{f[j]}$. Finally, the direction vector $\boldsymbol{D}$ must be reversed and Equation 2 is replaced by:

$$d_{ij} = V_{ij} \cdot (D_u) \tag{3}$$

In terms of performance, note that equations 2 and 3 avoid a length computation ($length = \sqrt{x^2 + y^2 + z^2}$) by using a scalar product approach. A straightforward step-by-step 3D-direction determination would need six sign determining steps; the scalar product method only needs a single sign determination.

The contact detection algorithm and contact reconstruction algorithm have been successfully implemented in C++. The system uses a publicly available collision detection module (RAPID[3]) and our reconstruction algorithm.[23] Rendering uses OpenInventor™, a commercial graphics toolkit. Users can build and edit objects using AutoCAD™ or any other method for generating DXF files, and third-party freeware converts AutoCAD™ DXF files to VRML/IV files, which the system inputs.

(ii) Experiment and Results. A test environment containing user-defined virtual fixed (static) and moving objects was built and tested. During the experiment, the user can manipulate (move/rotate) the objects using the mouse and keyboard, and see virtual collisions on the screen.

Figure 4 shows four test cases of varying difficulty ranging from a line collision to a multiple point collision of non-convex objects. The contact points edges and faces are highlighted in the figure (and on the screen). Detailed contact coordinate information is displayed to the user on the screen, along with the reversing vector. The algorithm's performance is shown in Table I. The experiment was run on

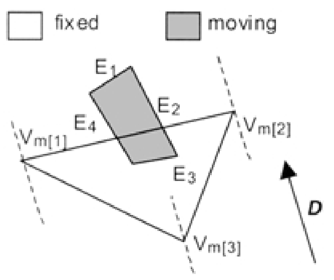

Fig. 3. Special situation for collision in 2D plane.

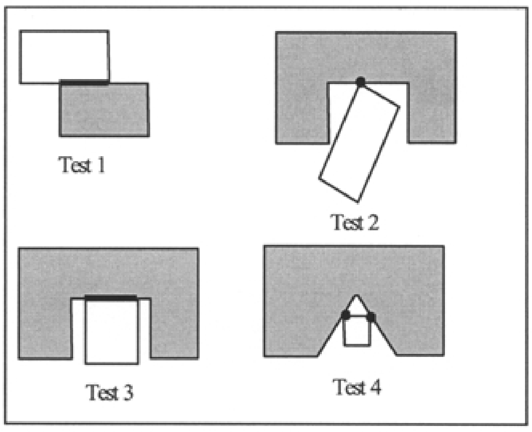

Fig. 4. Experimental test cases.

an NT workstation with a Pentium II-333 CPU and 128MB of memory. Figure 5 shows the contact reconstruction algorithm. Table I shows that for a specific case, how many lines and vertices are interfered and the time it takes for RAPID to determine these outcomes. Table 1 also shows the time it takes for reconstructing the contact configurations shown in the Figure 4.

3. INTERACTION WITH COMPLIANT OBJECT

This section presents an approach for enhancing graphical objects with physical behavior, specifically modeling objects that have compliant behavior. An example of this type of environment is the surgical environment where most objects in the surgical site have deformed under application of forces, for example from a surgical instrument.

The approach presented in this section is to represent the surface of the object with a massspringdamper system. To accomplish this, the surface of the model of the undeformed object is discretized into a triangular mesh via Delaunay triangulation.[24,25] Vertices of each triangle are then considered as lumped mass points; edges represent connecting springs and dampers in parallel. Depending on the local curvature of the surface the mesh can be finer or coarser. The nominal undeformed spatial position of each mass is considered the rest (or home position). Any displacement of a vertex is considered from this home configuration. When a vertex is displaced due to the action of applied contact forces, the restoring spring forces on connecting springs can be obtained. A haptic interface device for creating the sensation of reaction forces (for e.g. to the hand of a surgeon) can then produce this calculated force.

Assuming the position of each mass element is described with respect to a local frame of reference and using Newton's equation, the linear dynamic model of each of the mass-spring-damper system can be written as:

$$M\frac{d^2x}{dt^2}+C\frac{dx}{dt}+K(x)x=F_{ext} \tag{3}$$

where M, C, $K(x)$ are the mass, damping and associated stiffness matrices of the system. F_{ext} is the external force vector acting on the mass elements. An interpretation of Equation (3) is when a mass point is displaced from its equilibrium configuration due to some external force, the internal forces F_{int} due to the connected spring and dampers will force the mass point toward an equilibrium configuration through the acceleration of the point mass. Combining the external force vector F_{ext} with the internal force vector, i.e. $F_{int}=C\frac{dx}{dt}+K(x)x$, the following simplified form results:

Table I. Performance of the collision detection and contact reconstruction algorithms.

Example	1	2	3	4
No. colliding lines	32	54	409	1062
No. colliding vertices	22	31	205	502
RAPID time (msec)	0.15	0.31	991	5000
Reconstruction time (msec)	0.0	0.15	1.05	2.15

```
Initialize $V_{f[i]}$, $V_{m[i]}$, $E_{m[j]}$, $E_{f[j]}$.
Set velocity vector $\mathbf{D}$ for moving object and initialize $l_{m[i]}$ and $l_{f[j]}$. Initialize maximum distance $d_{max}$.
    // calculate $d_{maxf}$ for the fixed object:
    $\mathbf{P}_{ij}$ =Intersection ($E_{f[j]}$ , $l_{m[i]}$) ;
    If $\mathbf{P}_{ij}$ inside Line segment $E_{f[j]}$, keep the $\mathbf{P}_{ij}$ ;
        Else $\mathbf{P}_{ij}$ = NULL;
    $\mathbf{V}_{ij}$ = ($\mathbf{P}_{ij}$ - $\mathbf{V}_{m[i]}$) when Pij != Null;
    for (m=0; $m \le i$; m++) {
     for (n=0; $n \le j$; n++) {
        if $d_{ij} = \mathbf{V}_{ij} \bullet \mathbf{D} \ge 0$.
            {   if $d_{ij} \ge d_{maxf}$
                {$d_{maxf} = \mathbf{V}_{ij} \bullet -\mathbf{D}$ ;
                        Add $\mathbf{P}_{ij}$ to contact array;
                        Set reversing Vector $Vr_{ij}$;
                }
            }
        }
    }
    // calculate $d_{max}$ for moving object ($d_{maxm}$ ) similarly
    $d_{max}$ =max($d_{maxf}$ , $d_{maxm}$)
```

Fig. 5. Contact reconstruction algorithm.

$$M\frac{d^2x}{dt^2}=F=F_{ext}-F_{int}$$

Equation (3) must be evaluated numerically. A standard method uses an *explicit Euler integration scheme*. For example, the integration at a vertex location i can be written as:

$$v_i^{n+1}=v_i^n+F_i^n\frac{\Delta t}{m} \qquad (4)$$
$$x_i^{n+1}=x_i^n+v_i^{n+1}\Delta t$$

where F_i^n is the net force acting on vertex i with mass m at time step n, and v_i^n and x_i^n are the velocity and position of vertex i at time step n. Higher-order schemes such as Runge-Kutta can also be used for better numerical accuracy and for smooth solutions. The explicit method has been very popular because of its simplicity and ease of implementation. However, implementation of this scheme requires the size of the time step Δt be inversely proportional to the square root of the stiffness, i.e. the Courant condition.[26,27] The reason is that explicit integration, which many approaches use, cannot avoid large errors/offsets if the time step is large. The errors/offsets accumulate from step to step. Through n steps, small errors/offsets can grow large. This can be due to the assumption that the internal forces remain constant over a very large time-step which in turn may induce large changes in position. In general, stable behavior of the system can be achieved only over a very small time step. This is often referred to as problems with sets of stiff equations.

Another approach, which has been proposed to be better than the explicit method, is *implicit* Euler integration.[18,19] Here equation (4) is modified to become:

$$v_i^{n+1}=v_i^n+F_i^{n+1}\frac{\Delta t}{m} \qquad (5)$$
$$x_i^{n+1}=x_i^n+v_i^{n+1}\Delta t$$

where F_i^{n+1} is the net force acting on vertex i at time $n+1$. In this scheme, the position of vertex i at time step $n+1$ is reached through applying the force at that time step. The main task is now to compute an expression for the net force at time step $n+1$ as a function of the net force at time step n. One approach is based on the expansion of the net forcing function using a Taylor series approximation:

$$F^{n+1}=F^n+\frac{\partial F}{\partial x}\Delta^{n+1}x+\frac{\partial F}{\partial v}\Delta^{n+1}v \qquad (6)$$

where $\Delta^{n+1}(.)=(.)^{n+1}-(.)^n$ represents the backward difference operator, and $x\in R^{3i}$ is a vector of nodal displacements. Matrices $\frac{\partial F}{\partial x}$ and $\frac{\partial F}{\partial v}$ have the form of the stiffness and damping matrix representation. As such, they can be represented in a special form of matrix namely, sparse and banded. For example, for a one-dimensional grid with a constant spring between the vertices, the general form of the matrix can be written as:[28]

$$\frac{\partial F}{\partial x}=k$$

$$\begin{bmatrix} -1 & 1 & 0 & 0 & \cdot & 0 & 0 & 0 & 0\\ 1 & -2 & 1 & 0 & \cdot & 0 & 0 & 0 & 0\\ 0 & 1 & -2 & 1 & \cdot & 0 & 0 & 0 & 0\\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\ 0 & 0 & 0 & 0 & \cdot & 1 & -2 & 1 & 0\\ 0 & 0 & 0 & 0 & \cdot & 0 & 1 & -2 & 1\\ 0 & 0 & 0 & 0 & \cdot & 0 & 0 & 1 & -1 \end{bmatrix}$$

For the two and three-dimensional case, the form of the above matrix can be compounded in a block diagonal structure. Here, the size of the diagonal band increases.

Now, using Equation (6) in Equation (5), we have:

$$v^{n+1}=v^n+(F^n+K\Delta^{n+1}x+C\Delta^{n+1}v)\frac{\Delta t}{m}$$

Using the equivalent expression for $\Delta^{n+1}x=(v^n+\Delta^{n+1}v)\Delta t$, we can rewrite the above equation as:

$$v^{n+1}=v^n+(F^n+Kv^n\Delta t+K\Delta^{n+1}v\Delta t+C\Delta^{n+1}v)\frac{\Delta t}{m}$$

or:

$$\Delta v^{n+1}\frac{m}{\Delta t}(I-K\frac{\Delta t^2}{m}-C\frac{\Delta t}{m})=F^n+Kv^n\Delta t \qquad (7)$$

Then the implicit formulation for calculating the velocity vector is:

$$\Delta v^{n+1}=W(F^n+Kv^n\Delta t)\frac{\Delta t}{m} \qquad (8)$$

where:

$$W=\left(I-K\frac{\Delta t^2}{m}-C\frac{\Delta t}{m}\right)^{-1}$$

It can be seen that the differences between the explicit formulation for the velocity and Equation (8) is the added *weighting term W* (global filter) and the term $Kv^n\Delta t$. If $W=I$ and the term $Kv^nA\Delta t=0$, the update Equation (8) reduces to the *explicit* formulation. As mentioned before, the force vector F^n includes the expression for the nodal external force vector, the internal force due to the spring and the internai damping forces. The term $Kv^n\Delta t$ can be thought of as an artificial viscosity to further dampout nodal displacement.

Equation (8) can be rewritten in the following form to show the nature of the proposed integration method. Multiplying both sides by $\frac{m}{\Delta t}$, we get:

$$F^{n+1} = WF^{n}_{new} \tag{9}$$

where $F^n_{new} = F^n + Kv^n \Delta t$ is the new nodal force vector. The above equation suggests that the new nodal forces at step $n+1$ are obtained as weighted values of all the nodal forces at time step n. Reference [28] showed the complexity of calculating F^{n+1} was $O(N^2)$. The following proposed algorithm reduces the computational time considerably.[29]

From Equation (9), a force vector at a vertex i can be written as:

$$F_i^{n+1} = \sum_{j=1}^{N} w_{ij} F_j^n \tag{10}$$

where the w_{ij} are the elements of the W matrix. Thus, the net force at a vertex is the sum of N multiplications.

The matrix $W = I - K\frac{\Delta t^2}{m} - C\frac{\Delta t}{m}$ has the following properties:

(i) If $K\Delta t^2 + C\Delta t \ll I$, then the $N \times N$ matrix is equivalent to an identity matrix, i.e. $w_{ii} \approx 1$. This implies that for low stiffness or small time steps, for example, the updated force at the ith node at time step $n+1$ is a function of its own applied force at time step n.

(ii) If $K\Delta t^2 + C\Delta t \gg I$ a constant matrix results which implies that forces before and after are the same, in other words, rigid body translation of the collection of nodes.

In general, for the condition of the second case to apply, one requires either a large magnitude for the spring/damping parameters or a large time step. Neither case is practical for implementation. In a simplified implementation, we assumed a more realistic modification of the first case, namely $K\Delta t^2 + C\Delta t < I$ where in this case the relationship of equation (10) still applies but the update force at a given node is only a function of its immediate neighbors. To show this we would like to show that the inverse of a sparse band matrix is a sparse band matrix.

Let $W = A^{-1}$ where $A = \left(I - K\frac{\Delta t^2}{m} - C\frac{\Delta t}{m}\right)$. We would like to show that if A is sparse and banded matrix, its inverse W has the same property. One approach is to use the first order Neumann polynomial as:[30,31]

$$W = 2D^{-1} - D^{-1}AD^{-1}$$

where D is a diagonal matrix. As can be seen from the above, the inverse of A can be approximated as a sparse band matrix. Hence, one proposed simplification to the general update equation given in equation (9) is based on using only the links to the *immediate* neighbors.

Table II. Comparison of neighboring nodes' weighted sum and non-neighboring nodes' weighted sum.

Example	1	2	3	4
No. nodes	8	112	512	738
Sum of neighbor nodes weights	0.0	0.994	0.99/8	0.9983
Sum of non-neighbor nodes weights	0.0	0.006	0.002	0.0017

A numerical study was undertaken to verify the validity of the immediate neighbors simplification of the paper. Table II shows the comparison between using the full matrix representation of the weighting parameters (global filter) as compared to the immediate neighbors' method. The investigation was carried for meshes of 8, 112, 512 and 738 nodes. As Table II indicates, the sum of the contributions from non-neighbor nodes has a very small contribution compared with the weight of immediate neighbor nodes. In all of these examples, the matrix W is constructed once when the models were built and then used throughout the on-line update computation.

4. EXPERIMENTAL RESULTS

The proposed approach for modeling deformable objects was evaluated using a 2-dimensional planar model, though the method was implemented in 3D. Initial object geometry was constructed using a widely available Computer-Aided-Design system (AutoCAD). Mesh generation was via a Delaunay triangulation algorithm.[24] The software was run on a Pentium II-333 MHz CPU with 128MB of memory.

Figure 6 shows snapshots of a 517 node object with a single step function external force applied to it. Figure 7 shows an example of a 790 node object with a constrained boundary. For this object, the combined time for the simulation and graphics resulted in a frame rate of 1 f/sec. This figure also shows the example where multiple external inputs can be included into the global deformation solution.

5. CONCLUSIONS

Inclusion of physics in the graphical representation of objects is becoming one of the key requirements in the development of virtual environments and real-time simulations. A particularly important aspect of the evolving field of the design of user environments is the notion of haptic feedback where through special-purpose mechanical devices; the user can interact with graphical objects and directly feel reaction forces. In addition, in simulating physical object forces, for example in the development of animated characters or a game environment, graphical objects should be constructed such that they behave based on the laws of physics.

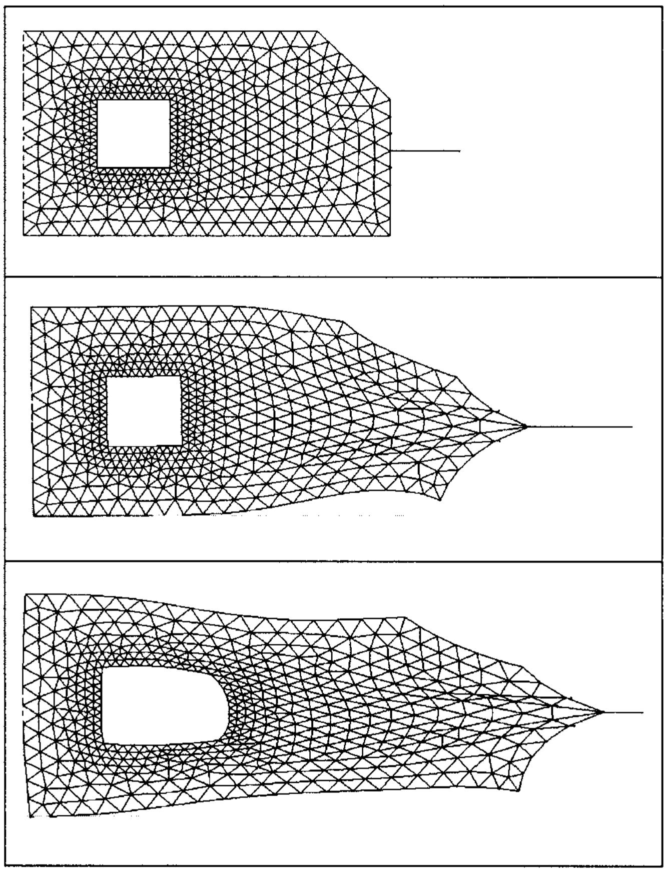

Fig. 6. Thin polygonal plate deformed by single step function force. Object has 517 nodes. Frames shown at $t=0$, 5s and 10s.

This paper presents some preliminary investigations into the design and implementation of such environments. Two aspects of such development were considered, namely (a) determinations of exact contact locations between two colliding objects and (b) an approach for modeling compliant objects and numerically solving such models. Determination of the exact contact location and configuration is one of the main requirements for further analysis of the behavior of objects. Having a point, line or surface contact between two objects can further allow proper dynamic model construction based on the initial conditions of the input forces. For example, in planning the motions for assembling two objects, such contact infornation can help the designer to predict possible motions that need to be made in order to assemble the parts. Modeling compliant objects is an important requirement for developing virtual environments in which most objects are soft, e.g. a surgical environment. An efficient framework needs to be identified which can result in an approach both realistic and computationally feasible. The proposed approach is based on representing the surface of the object with a 2-dimensional mesh with a system of mass-spring-damper. An efficient implicit numerical scheme was proposed for solving the deformation of such a mesh under the action of external forces. The proposed method utilizes a global filter where the local deformation can be mapped to the neighboring ones for more distributed and realistic model of the deformation.

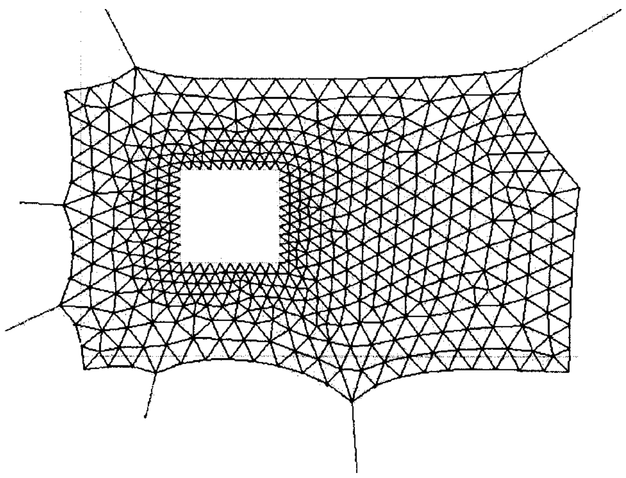

Fig. 7. Simulation of a deformable object with 790 nodes, 6 external forces and a constrained internal boundary.

References

1. E. G. Gilbert, D. W. Johnson and S. S. Keerthi, "A fast procedure for computing the distance between objects in three-dimensional space", *IEEE J. Robotics and Automation* **4**, 193–203 (1985).
2. D. Bara, "Curved surfaces and coherence for non-penetrating rigid body simulation", *Proc. Siggraph '90, Computer Graphics* **24**, No. 4, 19–28 (1990).
3. M. C. Lin and F. C. Johy, "Efficient algorithms for incremental distance computation", *IEEE Conference on Robotics and Automation* (1991), pp. 1008–1014.
4. S. Gottschalk, M. Lin and D. Manoch, "Obb-tree: A hierarchical structure for rapid interference detection", *Proc Siggraph '96* (1996), pp. 171–180.
5. G. Barequet, B. Chazelle, L. Guibas, J. Mitchell and A. Tal, "Box-tree: A hierarchical representation of surfaces in 3D", *Proceedings of Eurographics* **15**, No. 3, 335–342 (1996).
6. S. Cameron, "Approximation hierarchies and s-bounds", *Proceedings Symposium on Solid Modeling Foundations and CAD/CAM Applications* (Austin, TX (1991), pp. 129–137).
7. M. A. Ganter and B. P. Isarankura, "Dynamic collision detection using space partitioning", *Proceedings of ASME Design Automation Conference* (1990), pp. 175–181.
8. S. F. Frisken Gibson, "Beyond volume rendering: Visualization, haptic exploration, and physical modeling of voxel-based objects", *Technical Report* (Mitsubishi Electric Research Laboratories, Cambridge Research Center, number 95-04, 1995).
9. M. Lin and J. Canny, "A fast algorithm for incremental distance calculation", *IEEE International Conference on Robotics and Automation* (1991), pp. 2670–2675.
10. B. Mirtich, "V-clip: Fast and robust polyhedral collision detection", *ACM Transaction on Graphics* **13**, No. 3, 177–208 (1998).
11. M. Lin and S. Gottschalk, "Collision detection between geometric models: A survey", *Technical Report* (Department of Computer Science, North Carolina University, 2000).
12. L. E. Malvern, *Introduction to the Mechamcs of a Continuous Medium* (Prentice Hall, 1969).
13. O. C. Zienkiewicz, *The Finite Element Method* (McGraw-Hill, 1977).
14. S. Cotin and H. Delingette, "Real-time surgery simulation with haptic feedback using finite elements," *Proceedings of 1998 IEE, International Conference on Robotic & Automation* (May, 1998), pp. 3739–3744.
15. H. Qin and D. Terzopoulos, "D-NURBS: A physics-based framework for geometric design," *IEEE Transactions on Visualization and Computer Graphics*, No. 1, 85–96 (March, 1996).

16. Y. Zhuang and J. Canny, "Haptic interaction with global deformation", *Proceedings of IEEE Robotics and Automation* (2000), pp. 2428–2433.
17. D. L. James and D. K. Pai, "ArtDefo, accurate real time deformable objects", *Proceedings of IEEE Robotics and Automation* (1999), pp. 65–72.
18. G. Debunne, M. Desbrun, M. Cani and A. Barr, "Dynamic real-time deformation using space and time adaptive sampling", *Proceedings of SIGGRAPH* (2001), pp. 31–36.
19. S. Payandeh and N. Azouz, "Finite elements, mass-spring-damper systems and haptic rendering", *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation* (2001), pp. 224–230.
20. O. Astley and V. Hayward, "Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents", *Proceedings of IEEE Robotics and Automation* (1998), pp. 989–994.
21. G. S. P. Miller, "The motion dynamics of snakes and worms", *Proc. Siggraph '88, ACM Computer Graphics 22*, No. 4, 169–173 (1988).
22. X. Provot, "Deformation constraints in a mass-spring model to describe rigid cloth behavior", *Proc. Graphics Interface '95* (1995), pp. 147–154.
23. Z. L. Cai, J. Dill and S. Payandeh, "Haptic rendering: Practical modeling and collision detection", *IMECE99/ DSCD, Proceedings e ASME Virtual Environment and Teleoperator System Symposium* (1999), pp. 81–86.
24. R. S. Jonathan, "Triangle: Engineering a 2D quality mesh generator and delaunay triangulator", *First Workshop on Applied Computational Geometry* (Philadelphia, PA), Assoc. for Computing Machinery (1996), pp. 124–133.
25. C. M. Hofmann, *Geometric and Solid Modeling* (Morgan Kaufmann, San Mateo, California, 1989).
26. R. Courant and D. Hilbert, *Methods of Mathematical Physics, Vol. I* (Interscience, London, 1953).
27. L. Lapidus and G. F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Science* (Wiley, New York, NY, 1982).
28. U. Gudukbay, B. Ozguc and Y. Tokad, "A spring force formulation for elastically deformable models", *Computers and Graphics* **21**, No. 3, 335–346 (1997).
29. M. Desbrun, P. Schrode and A. Barr, "Interactive animation of structure deformable objects", *Proceedings of Computer Graphics Interface 1999* (June, 1999), pp. 1–8.
30. Z. L. Cai, J. Dill and S. Payandeh, "Haptic rendering: Toward deformation modelling with haptic feedback", *IMECEOO/ DSCD, Proceedings ASME Virtual Environment and Teleoperator System Symposium* (2000), pp. 1133–1138.
31. G. Meurant, *Computer Solution of Large Linear Systems* (Elsevier Science, 1999).