

Fast and Controllable Simulation of the Shattering of Brittle Objects

Jeffrey Smith
Robotics Institute
Carnegie Mellon University

Andrew Witkin
Pixar Animation Studios

David Baraff
Pixar Animation Studios

Abstract

We present a method for the rapid and controllable simulation of the shattering of brittle objects under impact. An object to be broken is represented as a set of point masses connected by distance-preserving linear constraints. This use of constraints, rather than stiff springs, gains us a significant advantage in speed while still retaining fine control over the fracturing behavior. The forces exerted by these constraints during impact are computed using Lagrange multipliers. These constraint forces are then used to determine when and where the object will break, and to calculate the velocities of the newly created fragments. We present the details of our technique together with examples illustrating its use.

Key words: Physically-based modeling, computer animation, impact, brittle materials.

1 Introduction

Realistic animation of breaking objects is difficult to do well using the traditional computer animation techniques of hand modeling and key-framing. This difficulty arises from the fact that the breaking of an object typically creates many small, interlocking pieces. The complexity and number of these fragments makes modeling them by hand all but impossible, but the distinctive look of a shattered object prevents the use of simple short-cuts, such as slicing the surface of an object into faces or the use of RenderMan shaders.

Consequently, the simulation of breaking and shattering has received some attention within the graphics community. An early attempt at modeling fracture is given in Terzopoulos and Fleischer[12], where they presented a technique for modeling viscoelastic and plastic deformations. While not specifically intended to model the breaking of brittle objects, their work allowed the simulation of tearing cloth and paper with techniques that could conceivably have been applied to this task. In 1991, Norton et. al. [7] described a technique specifically for modeling the breaking of three-dimensional objects wherein the object to be broken was subdivided into a set of equally-sized cubes attached to one another with springs. Unfortunately, their use of an elastic network invited massive

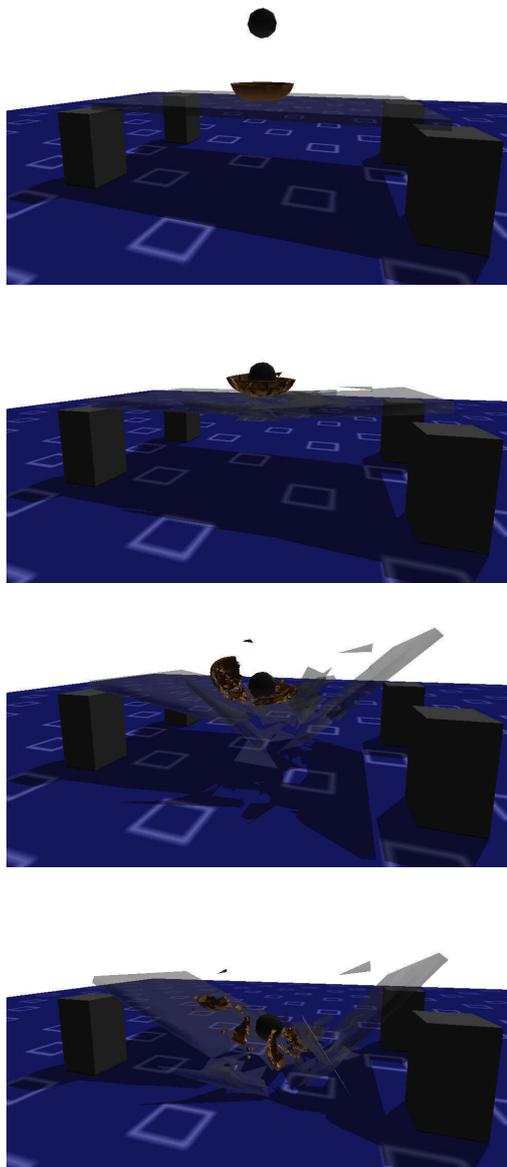


Figure 1: A bowling ball is dropped onto a ceramic bowl that is sitting on a thick glass table. Images are roughly 0.33 seconds apart

computational expense for large objects. Most recently, O'Brien and Hodgins[8] used continuum mechanics techniques developed in mechanical and civil engineering to model flexible objects, which included crack initiation and propagation. This method was rather slow, however, due to a combination of high physical realism and a complex system for dynamically re-meshing the solid during simulation.

Unsurprisingly, the fields of condensed-matter physics and materials science have examined this topic more thoroughly. Within the past decade many papers have been published on the subject of the simulation of brittle¹ fracture. A commonly used approach in these papers is the lattice model (Arabi[1], Chung[3], Donze[4]). This method models objects as a lattice of points or point-masses connected by stiff springs. During simulation, the extension of each spring, or some other potential function of the particle displacements, is computed. Depending on the model, either every element exceeding its extension or potential limit, or only the most egregious violator, is removed. The state of the system is cleared and the process repeated until the object falls apart, or until no new elements are being broken.

Although simple and general, the simulation methods outlined in the materials science literature are inappropriate for graphics applications. In general, materials scientists are interested in predicting with great accuracy how and when an object will shatter, whereas a computer animator is more concerned with generating realistic-looking behavior in a reasonable amount of time. Consequently, the level of physical detail used in materials science simulations is much higher, and often of a different nature, than we require.

A significant disadvantage of material science lattice solutions is that they almost universally use three-dimensional systems of stiff springs combined with explicit numerical integration methods. The step sizes for this type of simulation must be on the order of the inverse of the speed of sound in the material being simulated, and thus the computational expense can be high. (Even with the use of implicit integration methods, the simulation of a lattice of stiff springs entails a computational cost at least equal to that of Lagrange multipliers.) For example, in Chung[3], the simulation of an object with 2701 lattice-links is done in 137 time-steps, each taking 86 seconds on a 75MHz MIPS R8000, for a total of three and a half hours. A comparable simulation with our method on the same hardware would take roughly five and a half

¹The term "brittle," as used in materials science literature (and in this paper) means that the substance does not undergo significant plastic (reversible) deformation before breaking[2]. That is, a brittle object will not bend much under stress, but either will resist almost completely or break catastrophically.

minutes.

The continuum mechanics approach used by O'Brien and Hodgins[8] does not explicitly use a lattice of springs attached to point masses, but their method suffers from some of the same computational slow-downs. Specifically, their use of Euler and second-order Taylor integrators restricts the time step of simulation to extremely small values. Due to their re-meshing technique, the timing of the examples presented in their paper is difficult to directly compare with our own. As a rough comparison, their "wall #2" mesh, with a final total of 8275 elements, took an average of 1098 seconds of computation per simulation second on an 195 MHz R10000 processor. Running a similarly sized model with the same impact and fragmentation characteristics took roughly 90 seconds for a complete simulation with our technique.

In this paper, we present a fast and controllable method for simulating the fracture of brittle objects for animation. This method differs from the majority of the reviewed literature in that we use a system of point-masses connected by workless, distance-preserving constraints to represent the object, rather than a lattice of stiff springs. Our use of rigid constraints follows from an abstraction of brittle material properties and allows us to solve for the forces exerted by these elements during impact much more quickly than using explicit methods and an elastic mesh. We compute our solution by constructing a large, sparse, linear system which we solve using conjugate gradient methods. The constraint forces, once calculated, indicate when and where the object will break. This information is then used to construct the fragments of the broken object from the original geometry and to solve for the final linear and angular velocities of these bodies. In addition to advantages of speed, our system retains a great deal of user controllability while still yielding realistic-looking output, making it well-suited for use in animation.

2 Modelling

As mentioned above, our lattice method is roughly based on the elastic networks common in material science literature. However, instead of a three-dimensional mesh of springs we use a lattice of rigid constraints to connect point-masses. We are motivated in this choice by both speed considerations and the nature of ideal brittle materials.

Consider the naive system of points and springs. Since we are simulating brittle objects, these springs must be very stiff: stiff enough that no visible flexing (plastic deformation) takes place during high-momentum impact. As the brittleness of an object increases, the stiffness of these springs increase, and the displacements they under-

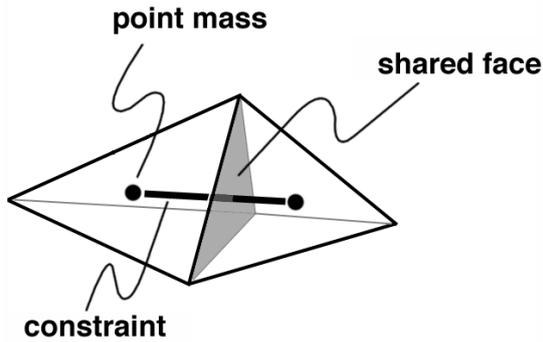


Figure 2: Two tetrahedra and their point/constraint complement

go during impact decrease. In the limit, then, for an ideally brittle material we would be forced to model springs which are infinitely stiff and undergo infinitesimal displacements.

Instead, we idealize these stiff springs as distance-preserving constraints and, rather than calculating displacements, calculate the forces that these constraints exert in response to an applied impulse. As mentioned earlier, a significant problem with simulating large systems of springs is the computational expense of explicit numerical integration. Our use of distance-preserving constraints allows a faster method of solution (discussed in section 3) while retaining both realism and a large amount of user-controllability.

2.1 Constructing the Model

The first step in our simulation is the construction of a solid model — consisting of a collection of simple polyhedra — from the initial description of the object to be shattered. Since this initial description is usually a set of points and faces describing only the surface, we first add a large number of well-distributed points inside this surface.

The addition of internal points is followed by a constrained Delaunay tetrahedralization. This operation yields our solid model, which consists of a set of connected tetrahedra. We then transform this solid model into a lattice representation — our final model. This lattice representation is simply the Voronoi complement (or “dual”) of the solid object, which represents the tetrahedra as point-masses and their connections (shared faces) as rigid constraints (figure 2). Each point-mass is placed where the center of the tetrahedra it represents was located and the mass of each point is determined by the volume of the tetrahedra and the density of the object in that volume.

The rigid constraints connecting these point-masses

have an associated ultimate strength which corresponds physically to the strength of the bond between two “micro-fragments” of the original object. If the tensile or compressive forces across a constraint exceeds this strength then the bond will be broken.

The breaking strength of a constraint is determined though a combination of user-specified functions and a pair of simple heuristics based upon the geometry of the two tetrahedra the constraint is “gluing” together. In our model, the strength of a constraint scales with the volume of the connected tetrahedra and with the area of their shared face. By relating the strength of a constraint to the size and shape of the tetrahedra it connects, the breaking behavior will be influenced by the geometry of the object, which is necessary for physically realistic results.

In addition to these simple geometry-based heuristics, the user may add a procedural variation to the constraint strengths. For example, simple cleaving planes may be added by systematically reducing the constraint strength along a cross-section of the object, or nodes of great strength may be created which will result in intact nodules remaining after the rest of the object is shattered. We have also achieved good results using noise and turbulence functions, as described in Perlin[9]. Much of the flexibility of our model comes from an appropriate choice of the function that determines the constraint strengths.

3 Simulation

Our approach to the simulation of fracture is a simple one, intended to avoid the computational expense and complexity of a full dynamic simulation while preserving physical realism. Although the time course of impacts can be as little as 100 microseconds, the speed of sound in brittle materials is typically several thousand meters per second². Given that the objects we wish to shatter are of moderate size (usually on the order of 10 cm on a side), the time to equilibrate internal forces (transmitted at the speed of sound) is on the order of one microsecond. Because the duration of a typical impact is so much longer than the time it takes the internal stresses to reach equilibrium, we make a quasi-static loading approximation, and can safely use global solution methods to calculate the forces between elements of the solid.

3.1 Fundamentals of the Simulation

We formulate the problem of calculating the forces being exerted by the rigid constraints as one of solving for Lagrange multipliers in the following simplification of the constraint force equation (For a derivation of this equation, see Witkin and Baraff[13] or Witkin, Gleicher and

²5100 meters/second in common glass and between 3500 and 7000 meters/second in hard stone.

Welch[14]):

$$JWJ^T\vec{\lambda} = -JWQ^T \quad (1)$$

where W is the inverse mass matrix and Q is the global force vector, containing information on what forces are being exerted on which particles by the impact.

The matrix J is defined as

$$J = \frac{\partial C}{\partial p}$$

where C is the ‘‘constraint vector’’: a vector of functions — one for each constraint in the system — whose values are zero if the constraint is being satisfied and non-zero otherwise. If we wish to introduce prior material stresses, the initial constraint vector may be given non-zero entries and equation 1 must then be changed to

$$JWJ^T\vec{\lambda} = -JWQ^T - kC$$

where k is some unit-normalizing factor.

Each constraint function is of the form

$$C_i(p_a, p_b) = \|p_a - p_b\| - d_i$$

where p_a and p_b are the locations of the two particles connected to constraint i , and d_i is the length of the constraint.

After solving equation 1 for $\vec{\lambda}$, we can calculate \hat{Q}

$$\hat{Q} = J^T\vec{\lambda}$$

which is the vector containing the forces being exerted by each constraint in reaction (and opposition) to the applied forces, Q . These values of \hat{Q} are then used to determine which constraints should be broken. Specifically, if a constraint is found to be exerting a force greater than its strength, it is removed. It should be noted that intergranular bonds in brittle materials are eight times stronger under compression than during extension [5], and this must be accounted for in our breaking decision-rule.

3.2 Physically Realizable Solutions

The system which we are solving:

$$JWJ^T\lambda = b \quad (2)$$

is underconstrained in the sense that for a given b , there are any number of $\vec{\lambda}$'s which satisfy the equation. However, an arbitrary vector $\vec{\lambda}$ does not necessarily correspond to a physically realizable set of constraint forces between connected particles. Given this fact, how can we be certain that our solution to equation 1 is the physically realizable one?

First we note that those solutions which are physically meaningful have a particular structure. Consider again

the connections between particles to be stiff springs. In this case, the only internal forces that can arise are those that have been generated due to some displacement δp of the particles. These displacements in turn correspond to a vector of spring tensions $\vec{\lambda} = J\delta p$. We can therefore see that all physically realizable $\vec{\lambda}$'s can be written as $\vec{\lambda} = J\delta p$ for some displacement δp . (We could parameterize by δp , but our solution would still have to satisfy equation 2 and our system would be more complex.) Stated a different way, any physically realizable $\vec{\lambda}$ must lie within the column-space of J and thus also in the column space of JWJ^T (regardless of J 's rank; see Strang[11] for details).

Note, though, that any solution $\vec{\lambda}$ of equation 2 that lies in the column space of JWJ^T is a minimum-norm solution. Thus, physically realizable solutions are equivalent to minimum-norm solutions, and since the minimum-norm solution to a linear equation is unique (Strang[11]), so is the physically realizable solution. Therefore, a solution method which finds a minimum-norm solution of equation 2 is guaranteed to give us the unique physically realizable solution $\vec{\lambda}$.

We use the conjugate gradient method to solve for the minimum-norm solution of our system. Not only will it give us the correct solution, as shown above, but it exploits the sparsity of the JWJ^T matrix to give us fast solution times[10].

3.3 Multiple-Step Solutions

It would appear that the simulation of an impact could be done with a single-step solution for \hat{Q} . However, our use of a global solution method would permit constraints to ‘‘transmit’’ forces of arbitrary strength before being removed, whereas we desire the constraints to be able to transmit no more force than their breaking strength would allow. Visually, a single-iteration solution results in the pulverization of a large volume surrounding the impact without the distinctive shards and fragments we desire.

Instead of a single iteration, however, we can solve for \hat{Q} in multiple steps, increasing the impact force with each iteration. In this way we can slowly ramp up the magnitude of the impact so that we are certain that no constraint transmits a force greater, to within some ϵ , than its breaking strength. By gradually increasing the magnitude of the impact force, we are impressing a pseudo time-course upon our simulation. That is, rather than simulating an impact as a single, zero-time impulse, we are creating a more realistic impact history. For all examples given in this paper, we used the simple piecewise linear function shown in figure 3 as our impact schedule.

Since, as mentioned before, the time to equilibrate the forces within a brittle object is much less than the duration of the impact we can safely chop this duration into

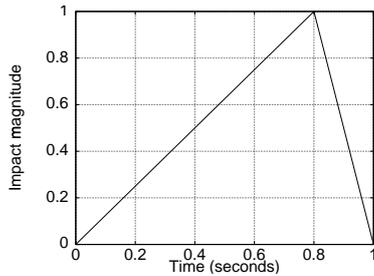


Figure 3: Impact magnitude versus time

smaller segments without losing the ability to solve with a global method. In practice, we have found that around 50 iterations of this loop generally yields acceptable results. Increasing the number of iterations beyond this brings little or no change in the fracturing behavior.

3.4 Crack Growth

Another important feature of brittle fracture that we would like to capture in our simulation is the growth of cracks. In brittle materials, the energy required to start a new crack of length l is significantly higher than the energy required to lengthen an existing crack by the same distance[6]. This behavior is the major reason why glass — despite its material homogeneity — breaks into large, polygonal shards under impact rather than turning into a cloud of tiny fragments.

In order to encourage the growth of pre-existing cracks, we modify our multi-step algorithm. When we remove a newly-broken constraint, we weaken the constraints around it that correspond to faces which adjoin the just-broken constraint. Thus, in the next iteration it is more likely that these constraints will break than constraints with an equal initial breaking-strength that are not connected to a pre-existing crack. By specifying what form this function takes the desirability of creating new cracks versus spreading existing flaws can be easily controlled.

To illustrate this effect, three examples were generated using the same model — a simple rectangular plank — the only difference between the simulations being the crack growth function used. The model used contained 3962 tetrahedra with 7096 shared faces. Constraint strengths varied between 90.2 and 541.0, having been generated with a combination of a turbulence function and the geometric heuristics described in section 2.1. Although these objects all have the same initial geometry and constraint values and are broken with the same impact, significantly different results were produced. Figure 5 shows (from the top) the aftermath of this test solid being fractured with no crack growth function. Here, dark lines show the edges of the top-facing tetrahedral faces and white lines indicate crack boundaries. We can

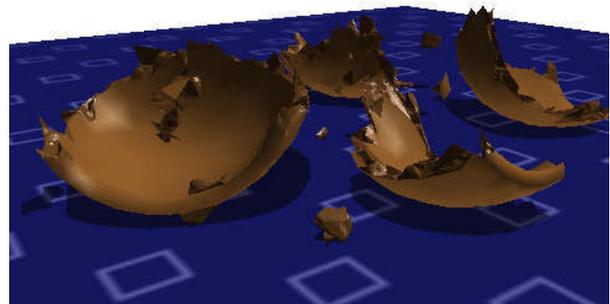
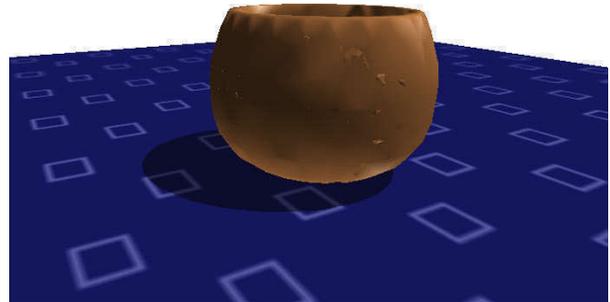


Figure 4: Glazed ceramic bowl, before (top) and after being broken with two different constraint-strength distributions

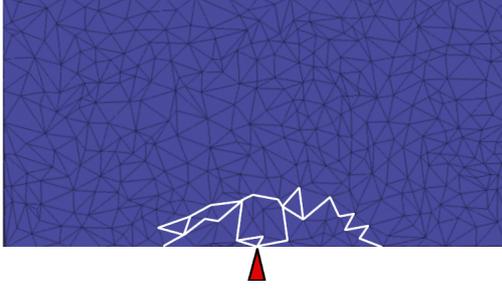


Figure 5: Top view of broken plank, showing cracks between tetrahedra

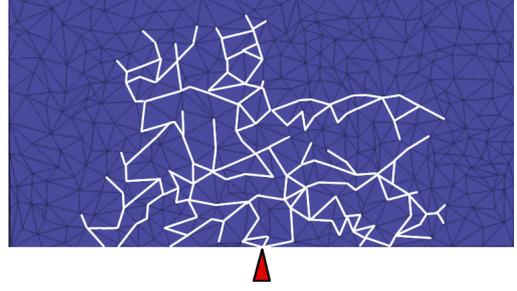


Figure 7: Top view of broken block, showing dramatic crack growth

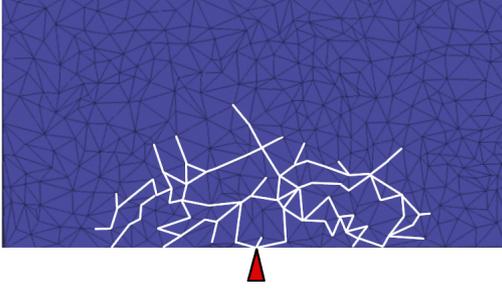


Figure 6: Top view of broken plank, showing moderate crack growth

see from this picture that the cracks which resulted in the fragmentation of this object have not spread far beyond the immediate impact location (the tip of the triangle).

Figure 6 shows the results of the same object being broken, but with a crack growth function that reduces constraint strengths by up to a factor of two. Specifically, the function

$$s_{new}^i = s_{old}^i (1.0 - \frac{1}{2} \sin(2\theta + \frac{\pi}{2}))$$

where θ is the angle between some constraint broken in the previous time-step and the neighboring constraint i (between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$). s_{old}^i and s_{new}^i are the old and new constraint strengths respectively. This crack growth function has encouraged the creation of more fragments, and has permitted parts of the object further from the impact site to break.

Finally, figure 7 shows the results our test object being broken again, but with a more extreme crack growth function. This function reduces constraint strengths by up to a factor of 1000:

$$s_{new}^i = s_{old}^i (0.5005 - 0.4995 \sin(4\theta + \frac{\pi}{2})) \quad (3)$$

Not surprisingly, cracks have propagated deeply into the solid and have caused it to break into many more pieces.

As can be seen from the three examples, even simple changes in the crack growth function can significantly alter our results, allowing the user further control over the material properties of the object.

3.5 Using the Results

Our algorithm produces as its output a large set of solids, each of which corresponds to a fragment of the original object. From the force of the impact and the individual masses of these fragments, we can easily compute their resulting angular and linear velocities.

Given a point i , we know that

$$\dot{p}_i = v_i + \omega_i \times p_i$$

where \dot{p}_i is the velocity of the point, v_i is the strictly linear velocity, ω_i is the angular velocity and p_i is its position. Thus, if we have a solid with three distinct points on its surface, p_0 , p_1 and p_2 , (trivial, since our most primitive solid is a tetrahedra), we can separate v_{solid} from ω_{solid} by solving the following simultaneous equation:

$$\begin{bmatrix} I & -p_0^* \\ I & -p_1^* \\ I & -p_2^* \end{bmatrix} \begin{bmatrix} v_{solid} \\ \omega_{solid} \end{bmatrix} = \begin{bmatrix} \dot{p}_0 \\ \dot{p}_1 \\ \dot{p}_2 \end{bmatrix}$$

where I is the 3 by 3 identity matrix and p^* is the dual matrix:

$$\begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix}$$

With these velocities in hand, we can perform a dynamic physical simulation to produce an animation of the aftermath of shattering.

4 Results and Discussion

We have described a simple, physically-motivated model for the rapid simulation of brittle fracture. The following examples illustrate the output of our work and demonstrate some of the different fracturing behaviors and material properties that can be simulated.

4.1 Wine glass

The examples shown in figure 8 were generated from the same geometric data: a wine glass modeled as 3422 tetrahedra with 6447 shared faces. Differences in fracturing behavior were produced by changing the function that determined the strengths of the constraints. More specifically, the constraint strengths were determined by a combination of a thresholded turbulence function and the geometric heuristics outlines in section 2.1. Each glass was broken with a single impact at the point where it struck the floor after falling.

4.2 Clay pot

Figure 4 shows before and after images of a pot, made from glazed earthenware. This model was constructed from 6902 tetrahedra, with 13150 shared faces. In the middle image, the initially homogeneous constraints were alternately strengthened and weakened along the vertical axis. This variation yields the characteristic breaking behavior of pottery created without a wheel, out of a single coil of clay. The lower image shows the same geometric model, but with constraints modified only by geometric factors and a mild turbulence function, which yields a very different set of fragments.

4.3 Glass table

Figure 1 shows a sequence of six images of a ceramic bowl, sitting on a thick glass table, broken by the impact of a falling bowling ball. For this example, the strength of the constraints in both broken objects (the bowl and the table-top) were homogeneous; modified only by the standard geometric factors. The crack growth function used in the table was that described in equation 3 which contributed to the formation of the long, narrow glass-like fragments.

4.4 Timing

Two major steps are involved in the destruction and subsequent animation of a shattered object: the impact calculation, and the reconstruction of the new fragments' surfaces afterwards. The left graph in figure 9 shows the amount of time required for each impact step calculation as a function of the number of constraints (shared faces) in the lattice model. (All timing was done on a 195 MHz R10000 SGI Octane.)

As can be seen, even for relatively large objects the impact simulation is computationally inexpensive. Since we are repeatedly solving a sparse linear system with the conjugate gradient method, our computational cost varies between $O(mn \log n)$ and $O(mn\sqrt{n})$ depending on the structure of JWJ^T .

Reconstruction of the fragments after impact requires similarly few resources. The right-hand graph in figure 9 shows the time required to construct the surfaces and ve-

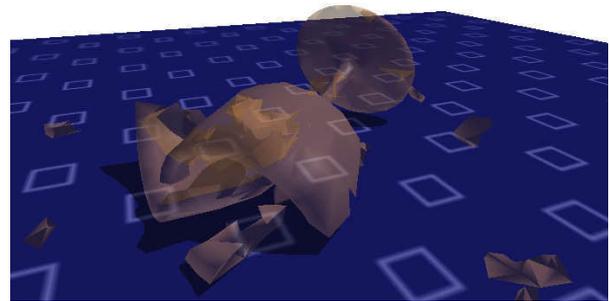
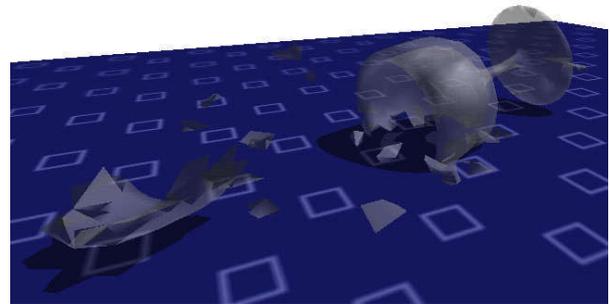
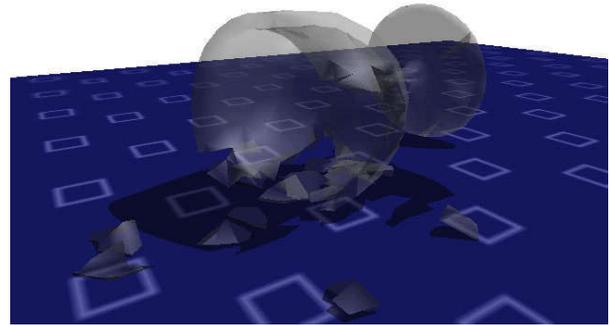


Figure 8: Three broken wine glasses, demonstrating different fracture behavior

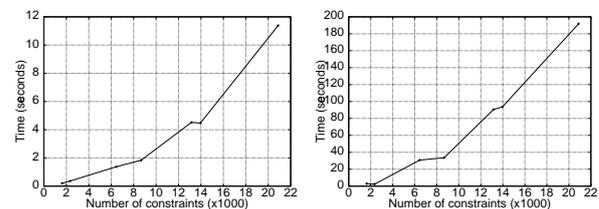


Figure 9: Time per impact step versus total number of constraints (left) and time required to construct fragments versus total number of constraints (right)

locities of the new fragments after impact. We find, then, that the total the time required to break and reconstruct models of moderate size (several thousand constraints) is only a few minutes.

5 Conclusion

We have presented a fast and controllable method for the simulation of the shattering of brittle objects. By framing the problem in terms of distance-preserving constraints rather than stiff springs, we have avoided expensive explicit solution methods while retaining physical accuracy. Furthermore, our method allows simple control over the ultimate number, size and shape of the fragments by adjusting the strength of the constraints throughout the body or by changing the nature of crack growth within the material. Combined with speed and accuracy, this controllability makes our method useful for the otherwise difficult task of animating complex, realistic shattering.

A significant limitation of our method, however, is the fact that, before fracturing behavior can be simulated, we must have a fully tetrahedralized model of the object to be broken. This requirement entails several disadvantages. First, the files describing the geometry of the object to be broken can become quite large, although compression of the text files, or storage in a compact binary form will offset this to a large degree. More significantly, a complete prior meshing means that fractures will only occur along mesh boundaries. Thus, if we wish to avoid meshing artifacts and “jaggy” fracture boundaries, the tetrahedralization must be done at a high resolution, which increases the computational expense (see figure 9). A solution to this shortcoming which we plan to pursue is to use a hierarchical approach along with fracture-boundary re-meshing such as that described in O’Brien and Hodgins[8]. This planned improvement, combined with more realistic crack-propagation functions, should substantially increase the realism of our method, which still allowing a great deal of user control over the fracture process.

References

- [1] S. Arbabi and M. Sahimi. Elastic properties of three-dimensional percolation networks with stretching and bond-bending forces. *Physical Review B*, 38(10):7173–7176, 1988.
- [2] G.P. Cherepanov. *Mechanics of Brittle Fracture*. McGraw-Hill, 1979.
- [3] J.W. Chung, A. Roos, and J. Th. M. De Hosson. Fracture of disordered three-dimensional spring networks: A computer simulation methodology. *Physical Review B*, 54:15094–15100, 21.
- [4] F. Donze and S.-A. Magnier. Formulation of a 3-d numerical model of brittle behavior. *Geophysical Journal International*, 122(3):709–802, 1995.
- [5] A.A. Griffith. The theory of rupture. *The Proceedings of The First International Congress of Applied Mechanics*, 1924.
- [6] B. Lawn. *Fracture of Brittle Solids*, chapter one: “The Griffith concept”. Cambridge University Press, 1993.
- [7] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney. Animation of fracture by physical modeling. *Visual Computing*, 7(4):210–219, 1991.
- [8] J. O’Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. *SIGGRAPH 99 Conference Proceedings*, 33:287–296, 1999.
- [9] K. Perlin. An image synthesizer. *SIGGRAPH 85 Conference Proceedings*, 19(3):287–296, 1985.
- [10] Jonathan R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CMU-CS-94-125, Aug. 1994.
- [11] G. Strang. *Linear Algebra and its Applications*. Harcourt Brace Jovanovich, 1988.
- [12] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *SIGGRAPH 88 Conference Proceedings*, 22:287–296, 1988.
- [13] A. Witkin and D. Baraff. *Physically Based Modeling: Principles and Practice*, chapter Physically Based Modeling. SIGGRAPH Course Notes, ACM SIGGRAPH, 1997.
- [14] A. Witkin, M. Gleicher, and W. Welch. Interactive dynamics. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, volume 24, pages 11–21, March 1990.