Interactive Deformation Using Modal Analysis with Constraints

Kris K. Hauser Chen Shen James F. O'Brien

EECS, Computer Science Division
University of California, Berkeley

Abstract

Modal analysis provides a powerful tool for efficiently simulating the behavior of deformable objects. This paper shows how manipulation, collision, and other constraints may be implemented easily within a modal framework. Results are presented for several example simulations. These results demonstrate that for many applications the errors introduced by linearization are acceptable, and that the resulting simulations are fast and stable even for complex objects and stiff materials.

Key words: Animation techniques, physically based modeling, simulation, dynamics, deformation, modal analysis, modal synthesis, finite element method, video games, interactive simulation, real-time simulation, constraints, precomputed dynamics.

1 Introduction

Interactive modeling of deformable objects has a wide range of applications from surgical training to video games. Many of these applications require realistic, real-time simulation for complex objects. Unfortunately, the most straightforward simulation methods turn out to be prohibitively expensive for modeling objects of even modest complexity. When the high cost of simulation couples with the reality that CPU cycles must be shared among many tasks, the need for faster, more sophisticated simulation methods becomes clear.

Recently several ingenious techniques for modeling deformable objects have been proposed. Examples include multi-resolution representations that avoid wasting time on irrelevant details (*e.g.* [4, 6, 8]), reformulating the dynamics to make them more stable (*e.g.* [15, 20]), extensive precomputation to minimize runtime costs (*e.g.* [9, 10, 19, 22]), robust integration schemes that afford large time-steps (*e.g.* [3]), and many other approaches that we cannot list here due to space constraints. As of yet, none provides a perfect solution that satisfies the requirements for all interactive applications.

This paper reexamines a technique known as modal analysis that was originally introduced to the graphics community over a decade ago, but has since been largely neglected, with only a couple of notable exceptions (*e.g.* [10, 22, 23]). Like the techniques mentioned above, modal analysis does not provide a perfect solution for every interactive application, but it does provide a solution that suits some applications quite well.

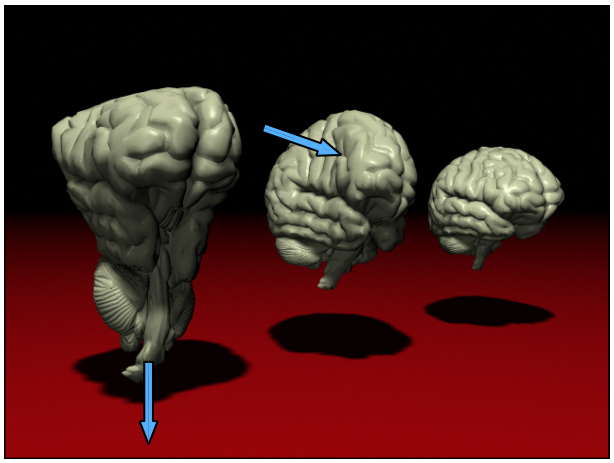

Figure 1: This example demonstrates a complex model being deformed using a modal simulation method. The object furthest from the viewer shows the undeformed configuration. The nearer objects are being deformed by a force indicated by the blue arrows.

The results presented here show that modal analysis can be used effectively to model situations where the deformable object is directly manipulated using constraints and where it interacts with an environment through contact forces. We demonstrate that although linear modal analysis does incur errors because of the inherent linearization of the dynamics, these errors are acceptable in many contexts, particularly when exaggerated cartoon-like deformations are desired. While precomputing the modal decomposition for a complex object may take up to a few hours of precomputation, for applications which make use of fixed content this computational cost only occurs during content development and it is well worth the dramatic increase in runtime performance.

The concepts required to manipulate the modal equations are to a certain extent conceptually difficult to work with but their implementation is surprisingly simple. The results shown in this paper (*e.g.* figure 1) were gener-

ated using an implementation that we have ported to several platforms: SGI IRIX, Windows, Linux, and Sony PlayStation2. On each of these platforms we were able to obtain interactive simulation times even for relatively complex models.

2 Background

Modal analysis is a well established mathematical technique that has been used extensively in mechanical, aerospace, civil, and other engineering disciplines for several decades. To a large extent the work we present in this paper follows as direct application of the methods developed in those fields to the task of interactively simulating deformable solids. There are, however, some issues that are unique to interactive simulation, such as imposing manipulation constraints and computing fast collision responses. This paper focuses on those issues. A discussion of modal analysis and its use with the finite element method can be found in the text by Cook, Malkus, and Plesha [5], and a more detailed discussion of modal analysis, its mathematical theory, and its applications may be found in the text by Maia and Silva [13].

Modal analysis was first introduced to the graphics community in 1989 by Pentland and Williams as a fast method for approximating deformation [19]. They used a hybrid framework, previously described by Terzopoulos and Fleischer [24], that separated the motion of a deformable solid into a rigid component and a deformation component. The deformable component existed in a non-inertial reference frame that moved with the rigid component. To avoid the cost of computing the modes for a particular object Pentland and Williams used linear and quadratic deformation fields defined over a rectilinear volume instead of the object's actual modes and then embedded the object within the region in a fashion similar to a free-form deformation. Although using approximated modes is computationally inexpensive, it only generates reasonable results for compact objects that are well approximated by a rectilinear solid. Pentland and his colleagues also integrated their modal deformation techniques into a interactive modeling system [18].

In 1997 Stam developed a modal method for modeling trees blowing in the wind [23]. Rather than starting with a deformable object, he computed the low-frequency modes from an articulated structure that described the tree. Once the closed-form solutions for each mode were computed, the response of the tree to a stochastic wind field could be computed efficiently.

Most recently, James and Pai implemented a system for computing real-time modal deformations on commodity graphics hardware [10]. They focused on modeling deformable skin and soft tissues attached to moving characters or as background elements in a surgical simulation. Shen and his colleagues have demonstrated an interactive system that could simulate models with over 10,000 vertices on a laptop PC with no special hardware acceleration [22].

Other related work includes sound generation techniques that make use of modal synthesis, and deformation techniques that use global shape functions that have some general similarities to a object's mode shapes. Van den Doel and his colleagues have used both analytically computed modes for simple geometric shapes and sampled modes from real objects to compute realistic sounds for simulated environments [26–28]. O'Brien and his colleagues developed similar techniques that used numerically computed modes from a finite element description of an object [17]. Examples of deformation techniques using global shape functions include: free-form deformations and their dynamics extensions [7, 21], deformable superquadrics [14], and the boundary element method [9]. Modal bases have also proven to be an efficient way to compactly encode both shapes and deformations [11, 12]. Finally, this paper focuses primarily on integrating manipulation and contact constraints into a modal framework, and there is prior work on applying these types of constraints to flexible body simulations [2].

3 Methods

The mechanical properties of an object can generally be captured by a function that maps the state of the object to a distribution of internal forces. For nearly any non-trivial system this function will be nonlinear and the representation of state will require many variables. Consequently, modeling the object's behavior over time will involve integrating a large, nonlinear system of differential equations. These systems are typically far too complex to be solved analytically, so some type of numerical solution method must be employed.

Modal analysis is the process of taking the nonlinear description of a system, finding a good linear approximation, and then finding a coordinate system that diagonalizes the linear approximation. This process transforms a complicated system of nonlinear equations into a simple set of decoupled linear equations that may be individually solved analytically.

The main benefit of this modal approach is that the behavior of the system can be computed much more efficiently. Because each of the decoupled equations can be solved analytically, the stability limitations that plague numerical integration methods are eliminated. Further, one may examine each of the decoupled components and discard those that are irrelevant to the problem at hand.

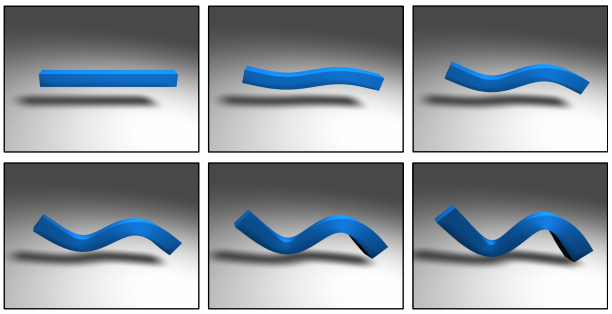

Figure 2: Using a linear formulation to model a bending bar produces acceptable results for small to moderate amounts of deformation. For larger deformations significant amounts of distortion appear. This example shows the deformation corresponding to the bar's second transverse mode.

There are also two drawbacks to a modal approach. First, linearizing the original nonlinear equations means that the solution will only be a first order approximation of the true solution. How objectionable the linearization error is depends on the application and the extent to which the objects deform from their initial configurations. As illustrated by figure 2, small to moderate deformations exhibit little or no noticeable error when casually observed. Even when the errors do grow noticeable, they have a cartoon-like, exaggerated appearance that may actually be desirable for some applications.

The second drawback arises because decoupling the linear system requires computing its eigendecomposition. However we do not believe that this drawback is particularly significant. The content in most interactive applications is constant, so that eigendecompositions can be precomputed during content development and stored with the objects. Furthermore, the linear systems are sparse, so that fast, robust, publicly available codes may be used to efficiently compute the decompositions (*e.g.* TRLAN [29]).

The remainder of this section describes how one computes the modal decomposition for a given object and how that decomposition can be used to efficiently model the object's behavior. Some of this material has been presented elsewhere by others in the graphics community (*e.g.* [10, 19]) but we include it here for completeness. The discussion will focus in particular on including manipulation and collision constraints in the modal framework. An overview of the entire process is shown in figure 3.

3.1 Modal Decomposition

The modal decomposition of a physical system begins with a linear set of equations that describe the system's behavior. In general, the equations describing the system

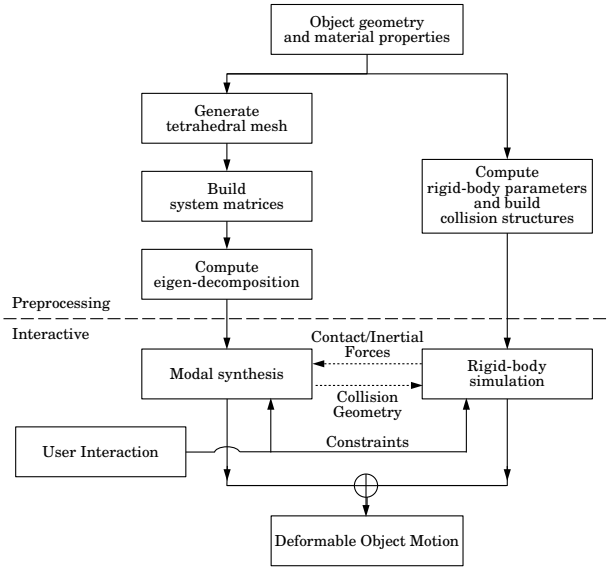

Figure 3: This diagram illustrates both the preprocessing steps used to construct the deformable modal model for an object, and the processes that subsequently generate interactive motion from this description.

may be nonlinear, and one obtains the linear equations by linearizing about some point, typically the rest configuration of the system. The linearized equations have the general form:

$$\boldsymbol{K}\boldsymbol{d} + \boldsymbol{C}\dot{\boldsymbol{d}} + \boldsymbol{M}\ddot{\boldsymbol{d}} = \boldsymbol{f} \ , \qquad (1)$$

where $\boldsymbol{K}$, $\boldsymbol{C}$, and $\boldsymbol{M}$ are respectively known as the system's stiffness, damping, and mass matrices, $\boldsymbol{d}$ and $\boldsymbol{f}$ respectively as the vector of generalized displacements and forces, and an overdot indicates differentiation with respect to time. The physical meaning of the generalized force and displacement vectors, and the method for computing the system matrices will depend on the type of method used for modeling the system. For general finite element methods, we refer the reader to the excellent text by Cook, Malkus, and Plesha [5]. We are using an implementation of the piecewise-linear tetrahedral finite element method described by O'Brien and Hodgins [16]. Details on computing the system matrices appear in [17].

Modal decomposition refers to the process of diagonalizing equation (1). The most general form of modal decomposition can be used for nearly arbitrary systems, but the systems arising from the finite element method we use have a structure that makes them amenable to a simpler manipulation provided we assume that the damping matrix, $\boldsymbol{C}$, is a linear combination of the $\boldsymbol{K}$ and $\boldsymbol{M}$. This restriction is known as Rayleigh damping, and although it is a restriction it still produces results superior to the simple mass damping that is most commonly used in

graphics applications. With these conditions, diagonalizing equation (1) becomes equivalent to solving a generalized symmetric eigenproblem with symmetric, positive-definite matrices. Cook, Malkus, and Plesha describe the process in detail and we only repeat the end result here.

With the restriction of Rayleigh damping equation (1) may be rewritten as:

$$\boldsymbol{K}(\boldsymbol{d} + \alpha_1 \dot{\boldsymbol{d}}) + \boldsymbol{M}(\alpha_2 \dot{\boldsymbol{d}} + \ddot{\boldsymbol{d}}) = \boldsymbol{f} \ , \tag{2}$$

where α_1 and α_2 are the Rayleigh coefficients. Let the columns of $\boldsymbol{W}$ be the solution to the generalized symmetric eigenproblem $\boldsymbol{K}\boldsymbol{x} + \lambda \boldsymbol{M}\boldsymbol{x} = 0$ and $\boldsymbol{\Lambda}$ be the diagonal matrix of eigenvalues[1], then equation (2) may be transformed to:

$$\boldsymbol{\Lambda}(\boldsymbol{z} + \alpha_1 \dot{\boldsymbol{z}}) + (\alpha_2 \dot{\boldsymbol{z}} + \ddot{\boldsymbol{z}}) = \boldsymbol{g} \ , \tag{3}$$

where $\boldsymbol{z} = \boldsymbol{W}^{-1}\boldsymbol{d}$ is the vector of modal coordinates and $\boldsymbol{g} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{f}$ is the external force vector in the modal coordinate system.

Each row of equation (3) corresponds to a single scalar second-order differential equation:

$$\lambda_i z_i + (\alpha_1 \lambda_i + \alpha_2)\dot{z}_i + \ddot{z}_i = g_i \ . \tag{4}$$

The analytical solutions to each equation are

$$z_i = c_1 e^{t\omega_i^+} + c_2 e^{t\omega_i^-} \tag{5}$$

where c_1 and c_2 are arbitrary (complex) constants, and ω_i is the complex frequency given by

$$\omega_i^{\pm} = \frac{-(\alpha_1\lambda_i + \alpha_2) \pm \sqrt{(\alpha_1\lambda_i + \alpha_2)^2 - 4\lambda_i}}{2} \ . \tag{6}$$

The absolute value of the imaginary part of ω_i is the frequency (in radians/second, not Hertz) of the mode, and the real part is the mode's decay rate. In the special case where the term under the radical in equation (6) is zero, we have $\omega_i^+ = \omega_i^-$, which gives the critically damped solution:

$$z_i = c_1 t e^{t\omega_i} + c_2 e^{t\omega_i} \ . \tag{7}$$

The columns of $\boldsymbol{W}$ are the vibrational modes of the object being modeled. (See figure 4.) Each mode has the property that a displacement or velocity over the object that is a scalar multiple of the mode will produce an acceleration that is also a scalar multiple of the mode. This property means that the modes do not interact with each other, which is why decoupling the system into a set of independent oscillators was possible. The eigenvalue for each mode is the ratio of the mode's elastic stiffness to the mode's mass, and it is the square of the mode's natural frequency (in radians per second). In general the eigenvalues will be positive, but for each free body in the system there will be six zero eigenvalues that correspond to the body's six rigid-body modes. The rigid-body eigenvalues are zero because a rigid-body displacement will not generate any elastic forces.

[1] Equivalently let $\boldsymbol{W} = \boldsymbol{L}^{-\mathsf{T}}\boldsymbol{V}$ where $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$ (Cholesky decomposition) and $\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{T}} = \boldsymbol{L}^{-1}\boldsymbol{K}\boldsymbol{L}^{-\mathsf{T}}$ (symmetric eigendecomposition).

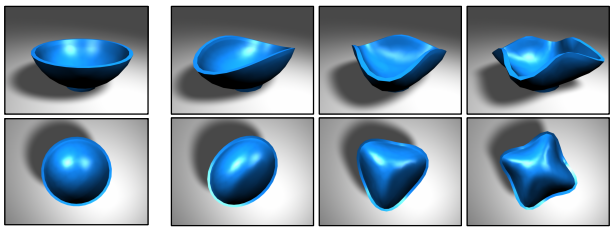

Figure 4: The two rows show a side and top view of a bowl along with three of the bowl's first vibrational modes. The modes selected for the illustration are the first three non-rigid modes with distinct eigenvalues that are excited by a transverse impulse to the bowl's rim.

The decoupled system of equations is *not an approximation* of the original linear system, it will generate exactly the same results as the original linear system. Of course the linear system may have been an approximation to some initial nonlinear one, but any problem that could be solved using equation (2) could also be solved with equation (3). Furthermore, simulation that would have required numerical time integration of equation (1) can now be solved without integration using the analytical solutions in equations (5) or (7).

3.2 Discarding Modes

Although decoupling equation (1) and then solving each of the resulting components analytically provides significant benefits, we can derive additional benefit by considering whether or not each of these components is needed. In particular we can discard modes that will have no significant effect on the phenomena we wish to model.

If the eigenvalue, λ_i, associated with a particular mode is large, then the force required to cause a discernible displacement of that mode will also be large. We can expect that in a given environment there will be both an upper bound on the magnitude of the forces encountered and a lower limit on the amplitude of observable movement. For example, if modeling an indoor environment we would not expect to encounter forces in excess of $60,000\,\mathrm{N}$ (the braking force of a large truck), and we would not be able to observe displacements less than about $0.1\,\mathrm{mm}$. Thus if $||\boldsymbol{w}_i||^2/\lambda_i < \mathrm{min\_res}/\mathrm{max\_frc}$ for some mode then that mode's behavior will be unobservable.

The imaginary part of ω_i determines the frequency that a mode will vibrate at. Modes that vibrate at more than half the display's frame rate will cause temporal aliasing.

Removing modes that are too stiff and/or too high frequency to be observed will not change the appearance of

the resulting simulation, but removing them will greatly reduce the simulation's cost. For most objects that we have worked with, nearly all of the modes are unobservable. A typical result is that an object with several thousand vertices will have many fewer than fifty modes that need to be retained. Furthermore, the number of modes that must be retained is nearly independent from the resolution of the model.

For later convenience let $\bar{\boldsymbol{W}}$ be the matrix $\boldsymbol{W}$ with the *columns* corresponding to the discarded modes removed, and let $\bar{\boldsymbol{W}}^{-1}$ be the matrix $\boldsymbol{W}^{-1}$ with the *rows* corresponding to the discarded modes removed. Note that $\bar{\boldsymbol{W}}^{-1} \neq (\bar{\boldsymbol{W}})^{-1}$, $\bar{\boldsymbol{W}}$ and $\bar{\boldsymbol{W}}^{-1}$ are not square, and $\bar{\boldsymbol{W}}^{-1}\bar{\boldsymbol{W}} = \boldsymbol{I}$ but $\bar{\boldsymbol{W}}\bar{\boldsymbol{W}}^{-1} \neq \boldsymbol{I}$.

3.3 Oscillator Coefficients and Time Steps

The analytical solution for each mode, equation (5), describes how that mode will behave when no external forces are acting on it. Using these solutions, however, requires some way of modeling responses to external forces and of setting initial conditions.

Given a set of initial conditions described by the node positions, $\boldsymbol{d}_0$, and their velocities, $\dot{\boldsymbol{d}}_0$, setting the oscillators to match those conditions requires finding appropriate values for the coefficients c_1 and c_2. First, the initial conditions are transformed to modal coordinates: $\boldsymbol{z}_0 = \bar{\boldsymbol{W}}^{-1}\boldsymbol{d}_0$ and $\dot{\boldsymbol{z}}_0 = \bar{\boldsymbol{W}}^{-1}\dot{\boldsymbol{d}}_0$. For each mode, c_1 and c_2 are given by

$$c_1 = \frac{z_0}{2} + \frac{(\alpha_1\lambda_i + \alpha_2)z_0 + 2\dot{z}_0}{2\sqrt{(\alpha_1\lambda_i + \alpha_2)^2 - 4\lambda_i}} \quad (8)$$

$$c_2 = \frac{z_0}{2} - \frac{(\alpha_1\lambda_i + \alpha_2)z_0 + 2\dot{z}_0}{2\sqrt{(\alpha_1\lambda_i + \alpha_2)^2 - 4\lambda_i}} \,. \quad (9)$$

For the critically damped case c_1 and c_2 are given by

$$c_1 = \frac{(\alpha_1\lambda_i + \alpha_2)z_0}{2} + \dot{z}_0 \quad (10)$$

$$c_2 = z_0 \,. \quad (11)$$

Note that if the $\omega_i^{\pm}$ are real then c_1 and c_2 will also be real. If the $\omega_i^{\pm}$ are complex then the $\omega_i^{\pm}$ and the c_1 and c_2 will be complex conjugate pairs. In either case equation (6) will evaluate to a real value.

To compute the response of a mode to an impulse delivered at $t = 0$, first transform the impulse to modal coordinates with $\Delta t\boldsymbol{g} = \Delta t\bar{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{f}$ and then compute c_1 and c_2 as shown above with z_0 set to zero and $\dot{z}_0$ replaced by $\Delta t g$. Because the modes behave linearly, the response of the system to forces applied at an arbitrary time may be computed by time-shifting this impulse response and adding it to the existing values.

Because $ce^{(t+\Delta t)\omega} = (ce^{t\omega})e^{\Delta t\omega}$, the state of each oscillator can be stored simply as a pair of complex numbers that reflect the current values of $c_1e^{t\omega^+}$ and $c_2e^{t\omega^-}$. Each time the system is advanced forward in time, these values get multiplied by $e^{\Delta t\omega^{\pm}}$. If Δt is constant then the step multiplier for each mode may be cached to avoid the cost of evaluating an exponential. Impulses applied to the system simply require adding the appropriate values to each oscillator's state. Finally, modes where ω^+ and ω^- are complex conjugate pairs (*i.e.* underdamped modes) can be reduced to only a single oscillator.

3.4 Constraints

Although we can compute the behavior of the decomposed system extremely efficiently, the method is not particularly useful unless it accommodates manipulation constraints and collision response. When working with the original system constraints on the node positions are nearly trivial to implement. Collision response requires more sophistication but still is conceptually straightforward. Unfortunately, applying these same constraints in the modal basis requires moving between the node positions and modal coordinates which can be unintuitive. Matters are further complicated because if we have discarded any modes then the transformations will be non-invertible.

3.4.1 Interactive Manipulation

If we wish to include continual constraints on part of the system, the optimal way to do so is to remove those degrees of freedom prior to performing the modal decomposition. Examples demonstrating this approach can be seen in James and Pai's modal method for modeling tissue deformation [10], and in our deformable sheet example. (See accompanying animations.) Using this approach for dynamic constraints, however, would require recomputing the eigendecomposition each time a constraint was added or removed from the system. James and Pai accomplished something similar for a boundary element method using Sherman-Morrison-Woodbury updates but we do not know of any corresponding incremental update scheme for an eigensystem [9].

Instead we apply manipulation constraints to the decomposed system. Let ψ be the set of degrees of freedom in the original system that we wish to constrain, and let ϕ be the places where we are willing to apply forces in order to enforce the constraints. For a manipulation task where a point on the object is being dragged we would typically have $\phi = \psi$ but we will not require it. Let $\boldsymbol{d}_\psi$ or $\boldsymbol{f}_\phi$ denote the displacement or force vectors where all except the elements corresponding to ϕ or ψ have been removed. Similarly, let $\bar{\boldsymbol{W}}_\psi$ be $\bar{\boldsymbol{W}}$ where all the rows not in ψ have been removed and let $\bar{\boldsymbol{W}}_\phi^{\mathsf{T}}$ be $\bar{\boldsymbol{W}}^{\mathsf{T}}$ where all the columns but for those in ϕ have been removed. Finally, let $\ddot{\boldsymbol{d}}_\psi^*$ be the desired accelerations at the constraint locations. By combining $\ddot{\boldsymbol{d}} = \bar{\boldsymbol{W}}\ddot{\boldsymbol{z}}$, $\boldsymbol{g} = \bar{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{f}$ and a bit

of manipulation we obtain:

$$\ddot{\boldsymbol{d}}^*_\psi = \bar{\boldsymbol{W}}_\psi(\ddot{\boldsymbol{z}} + \bar{\boldsymbol{W}}^{\mathsf{T}}_\phi \boldsymbol{f}_\phi) \; . \tag{12}$$

Solving for $\boldsymbol{f}_\phi$ yields:

$$\boldsymbol{f}_\phi = \left(\bar{\boldsymbol{W}}_\psi \bar{\boldsymbol{W}}^{\mathsf{T}}_\phi\right)^{-\mathsf{P}} \left(\ddot{\boldsymbol{d}}^*_\psi - \bar{\boldsymbol{W}}_\psi \ddot{\boldsymbol{z}}\right) \; , \tag{13}$$

where $\cdot^{-\mathsf{P}}$ denotes a pseudoinverse. Velocity constraints only differ in that $\boldsymbol{f}_\phi$ gets replaced by an impulse, *e.g.* $\Delta t \boldsymbol{f}_\phi$ and we have:

$$\boldsymbol{f}_\phi = \frac{1}{\Delta t}\left(\bar{\boldsymbol{W}}_\psi \bar{\boldsymbol{W}}^{\mathsf{T}}_\phi\right)^{-\mathsf{P}} \left(\dot{\boldsymbol{d}}^*_\psi - \bar{\boldsymbol{W}}_\psi \dot{\boldsymbol{z}}\right) \; . \tag{14}$$

Position constraints can be enforced in a similar fashion so long as we adjust for how each mode will evolve over the interval while the force is applied:

$$\boldsymbol{f}_\phi = \frac{2}{\Delta t^2}\left(\bar{\boldsymbol{W}}_\psi \boldsymbol{S} \bar{\boldsymbol{W}}^{\mathsf{T}}_\phi\right)^{-\mathsf{P}} \left(\boldsymbol{d}^*_\psi - \bar{\boldsymbol{W}}_\psi \boldsymbol{z}\right) \; , \tag{15}$$

where $\boldsymbol{S}$ the diagonal matrix with components given by

$$s_{ii} = \frac{e^{\Delta t \omega_i^+} - e^{\Delta t \omega_i^-}}{|\sqrt{(\alpha_1 \lambda_i + \alpha_2)^2 - 4\lambda_i}|} \tag{16}$$

that compensates for the motion of each mode during the interval.

3.4.2 Dynamics Simulation

Implementing a deformable dynamics simulator for free bodies using modal analysis can be accomplished by combining the modal simulation with a standard rigid-body dynamics simulator. The modal system is embedded in a rigid-body reference frame, and both systems evolve over time. The two systems interact with each other though inertial effects. The modal system should experience centrifugal and coriolis forces as the rigid-body moves, and the inertial moments of the rigid-body will change as the modal system deforms. Unless the object is rotating rapidly, neither effect will be significant so we omit them. They could be included at an additional computational cost. Inertial effects due to translational and rotational acceleration of the rigid-body frame do not need to be modeled explicitly so long as the forces generating those accelerations are also applied to the modal system.

Because we are modeling deformable objects, a collision detection method optimized for use with rigid-body simulations requires some modification because precomputed data structures will become invalid as the object deforms. The method we are using employs a hierarchy of axis-aligned bounding boxes, aligned to the world axes, to efficiently find potential collisions. The tree is initially constructed based on the undeformed shape of the object. Each leaf node in the tree corresponds to one of the primitives that makes up the object, and the bounding box at that node encloses the primitive. The bounding boxes of interior nodes encompass the union of their children. The tree's topology is chosen to minimize the overlap among the interior nodes. Once the object deforms the tree will become invalid, but recomputing the tree's topology every time-step would be prohibitively expensive. Instead we use an update scheme similar to one described by van den Bergen [25]. After each time-step the bounding boxes are updated, but the tree's topology does not change. If we expected arbitrary deformation, this could result in a very poorly structured tree, but because the extent of deformation is limited we have found this approach to work quite well.

Using these trees the collision system can efficiently determine contact points and a normal for each contact. For collisions between an object and a ground plane, the collision normal is simply the plane's normal. For collisions between objects, we look at involved tetrahedra to determine a normal based on their overlap [16]. We have found that each physical contact site may produce several pairs of colliding primitives. To reduce the computation when using constraint-based collisions we cluster nearby collision points and treat each cluster as a single collision point.

We have implemented collision response using both a penalty-based method and using constraints. As one would expect, the penalty methods require less work per time-step, achieving real-time performance, but stiff penalty coefficients can lead to instability. The constraint-based method requires more work per time-step, but it is more stable. Because the modal system will allow arbitrarily large time-steps in the absence of external influences we prefer the more stable constraint-based methods.

To implement penalty methods, when a point on a surface violates one of the penalty constraints, a force proportional to the magnitude of the violation is applied at that point. Transforming the forces to modal coordinates and then applying the force to the modal system is done as described previously. The penalty force should be applied to both the modal and the rigid-body systems.

Constraint-based collisions require a more complex implementation, but we find that they produce better results. First, when a collision occurs, the simulation is backed up to the point during the time-step when the objects first came into contact. Then contact forces are calculated as the minimal outward normal force to ensure that the objects will not continue to penetrate. These are determined by solving a linear programming problem for the normal forces at all contact points. Baraff details an efficient method for solving for the required forces [1].

Constraint methods are often used in traditional rigid-body simulations only to solve for resting contact, while

impulses are used to calculate elastic response. Elastic components of the response can be handled differently in our modal simulation, because the elastic behavior of the modal system models them directly. We first enforce a velocity constraint that solves for an impulse to ensure that none of the contact velocities are negative, then secondly it enforces an acceleration constraint that solves for a force to ensure that none of the contact accelerations are negative. The derivation of these methods requires equations relating the change in velocity and acceleration with respect to an applied impulse and acceleration, respectively.

Let $\boldsymbol{p}_l$ be the location of a contact point on an object expressed in the local coordinate frame of the rigid body. This location will be a linear function of the modal coordinates so that:

$$\boldsymbol{p}_l = \boldsymbol{U}\boldsymbol{W}\boldsymbol{z} \ , \tag{17}$$

where $\boldsymbol{U}$ is a matrix that averages the appropriate node locations based on the barycentric location of $\boldsymbol{p}$ in one of the surface triangles. The location in world coordinates is given by

$$\boldsymbol{p}_w = \boldsymbol{t} + \boldsymbol{R}\boldsymbol{p}_l \ , \tag{18}$$

where $\boldsymbol{t}$ and $\boldsymbol{R}$ are the translation and rotation matrices for the rigid-body frame. Differentiating with respect to time to obtain the world velocity and acceleration of $\boldsymbol{p}$ yields:

$$\dot{\boldsymbol{p}}_w = \dot{\boldsymbol{t}} + \boldsymbol{R}[\boldsymbol{\omega}]\boldsymbol{p}_l + \boldsymbol{R}\dot{\boldsymbol{p}}_l \ , \tag{19}$$

$$\ddot{\boldsymbol{p}}_w = \ddot{\boldsymbol{t}} + \boldsymbol{R}[\boldsymbol{\omega}][\boldsymbol{\omega}]\boldsymbol{p}_l + \boldsymbol{R}[\boldsymbol{\alpha}]\boldsymbol{p}_l + 2\boldsymbol{R}[\boldsymbol{\omega}]\dot{\boldsymbol{p}}_l + \boldsymbol{R}\ddot{\boldsymbol{p}}_l \ , \tag{20}$$

where $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ are the rigid-body's angular velocity and acceleration[2]. The notation $[\boldsymbol{a}]$ denotes the skew-symmetric matrix such that $[\boldsymbol{a}]\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b} = -[\boldsymbol{b}]\boldsymbol{a}$.

Differentiating equation (19) with respect to an applied impulse allows us to obtain the change in velocity generated by a constraint force over a time interval:

$$\Delta\dot{\boldsymbol{p}}_w = \Delta t \left(\frac{1}{m}\boldsymbol{f}_w + \boldsymbol{R}[\boldsymbol{H}^{-1}\boldsymbol{\tau}_l]\boldsymbol{p}_l + \boldsymbol{R}\boldsymbol{U}\boldsymbol{W}\bar{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{f}_l \right) \tag{21}$$

where $\boldsymbol{H}$ is the object's inertia matrix and $\boldsymbol{\tau}$ is the torque generated by $\boldsymbol{f}$. Differentiating equation (20) with respect to an applied force produces a similar result for the change in acceleration at the contact point. These equations are linear in $\boldsymbol{f}$, and can be used similarly to solve for position, joint, and collision constraints. Position constraints require that a point's velocity and acceleration are zero. Joint constraints require relative velocities and accelerations are zero, merely requiring a subtraction of the proper terms. Collision constraints require the normal components of relative velocities and accelerations are nonnegative, and only solve for the nonnegative normal force magnitude. All constraints are solved simultaneously as a linear program. Solving cannot always be done in real-time if there are a large number of contact points, although system response does remain interactive.

[2]In order to adhere to common convention we are reusing ω and α, that were previously used for the modal frequencies and Rayleigh damping coefficients. The intended meaning should be clear from context and the presence/absence of bold notation.

Example	Fig	Verts.	Nodes	Tets.	Modes	Time
Brain	1	18,847	304	997	40	68.5 sec
Dodo	5	336	113	295	40	6.2 sec
Bunny	8	2,633	37,114	15,507	32	24 min
Sphere	video	66	80	282	40	2.9 sec
Sheet	video	195	195	486	20	14.4 sec
Bat	video	241	310	1,030	20	68.9 sec

Table 1: This table list the number of vertices in the rendered models, the number of nodes and elements in the finite element models, the number of modes retained, and the time required to compute the decomposition for some of the demonstration objects.

We model friction at the contacts using a simplified Coulomb friction model. The system computes a force opposite the tangential velocity at the contact points. The magnitude of the force equals the magnitude of the normal force multiplied by a friction coefficient. If the friction force causes the predicted tangential velocity to be reversed then it is limited to the force that would cause no slipping. If interactivity can be sacrificed, a more precise method would be to add an additional no-slip constraint to be re-solved with the other constraints. We find our heuristic reasonable for producing plausible friction effects.

4 Results

We have implemented a system that models deformable objects using a hybrid formulation that combines rigid-body motion with deformation computed using modal analysis. Objects may be interactively manipulated by the user with both penalty forces and displacement constraints. The modal objects may collide with each other and with their environment. Collisions can be treated with either penalty forces or constraints, and objects may also be attached together using joint constraints. Table 1 lists several of the models we have used to demonstrate our results and shows the geometric and kinematic complexity of the models along with how much precomputation time was required to perform the modal decomposition for each model.

The brain model in figure 1 demonstrates pulling and pushing using force application. Force vectors are projected into the modal basis, modifying the modal state, and then are projected out, resulting in realistic deformation. The images in figure 6 and figure 7 show pulling and pushing using manipulation constraints. Typically, up to

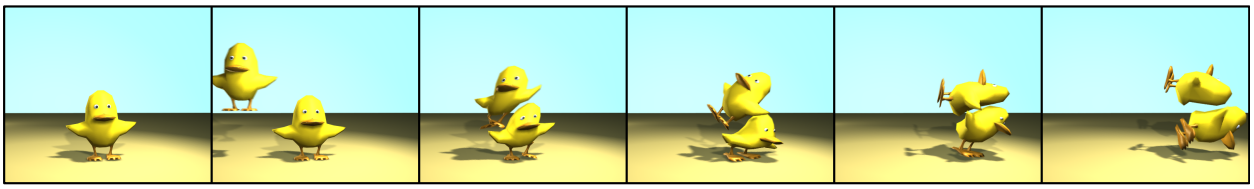

Figure 5: This image sequence shows frames from an animation of a pair of objects colliding with each other. Each object is a hybrid simulation that incorporates a rigid and a deformable (modal) component.

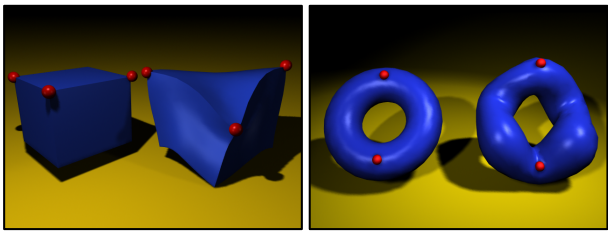

Figure 6: These images shows how constraints can be used to deform objects. The object on the left of each image shows the object prior to deformation, and the right object shows the results after the red constraint points have been moved.

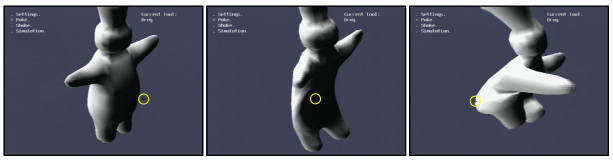

Figure 7: These images are screen shots from an application running natively on a Sony PlayStation2. The yellow circle highlights the cursor that the user is using to poke and pull an elastic figure.

around 10 points on the model can be constrained in real-time on a moderate speed computer (300 MHz Pentium II or Sony Playstation2). A limit is reached because the solutions to equation (13) and equation (15) require a relatively expensive computation of singular value decompositions, which cannot be calculated in real-time once the matrices become too large.

We have created several animations (see supplemental materials) demonstrating this system, each simulated interactively for moderately complex objects. The results appear plausible, and resemble animations that might be simulated using more straightforward but more computationally expensive methods. The bottlenecks in hybrid modal/rigid-body simulation are collision detection and solving the linear program for the constraints. To reduce the computation used in solving the linear program, the extent of contact point clustering may be tweaked to sacrifice accuracy for speed. Figures 5 and 8 show objects involved in collisions with a ground plane and each other.

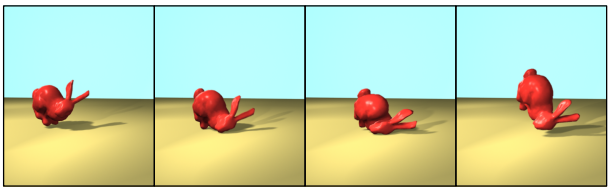

Figure 8: A sequence of images showing the Stanford Bunny model bouncing off a ground plane.

As with other methods based on tetrahedral finite elements, we can embed high-resolution or non-manifold surfaces inside a tetrahedral volume model. The benefits of this technique are that the surface shading and texturing can be specified independently from the dynamics, and poorly constructed "polygon-soup" models may be used. Both the brain model in figure 1, an extremely complex object, and the "dodo" model in figure 5, a non-manifold object, are modeled in this way. The "dodo" model also demonstrates non-uniform material properties: the legs and beak are made of a stiffer material than the rest of the body.

5 Conclusions

Modal analysis has been shown to be a useful tool for interactively producing realistic simulations of elastic deformation. Both the analytic calculation of modal amplitudes using complex oscillators and the removal of high-frequency modes have a stabilizing effect on simulations, allowing for large time steps to be taken.

Despite the approximation of linearity in modal analysis, the simulation results are quite plausible for most objects. The exceptions are long, thin, or highly deformable objects, where nonlinear behavior dominates the expected behavior. Despite these specific drawbacks, many objects can be manipulated quite efficiently and realistically using modal models.

The already small costs of modal analysis can be reduced further by leveraging graphics hardware, as shown by James and Pai [10] or our own implementation on the Sony PlayStation2. Using such hardware, CPU costs can be reduced to modifying mode amplitudes during evolution of time steps, projection of forces, and application of manipulation constraints.

We recognize that there are many implementation details that cannot fit into this paper, so we have released the source code for our Linux implementation under the GNU License. It is our hope that making this code available will encourage others to work with modal simulation methods. The code may be accessed at www.cs.berkeley.edu/∼job/Projects/ModeDef.

Acknowledgments

The authors thank Christine Gatchalian for her help with the models used in the example, the other members of the Berkeley Graphics Group for their support, and the reviewers for their insightful comments.

This work was supported with contributions from Sony Computer Entertainment America, Intel Corporation, and Pixar Animation Studios, and with NFS grant CCR-0204377 and State of California MICRO grant 02–055.

References

[1] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of SIGGRAPH 94*, pages 23–34, July 1994.

[2] David Baraff and Andrew P. Witkin. Dynamics simulation of non-penetrating flexible bodies. In *Proceedings of SIGGRAPH 92*, pages 303–308, July 1992.

[3] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[4] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. In *Proceedings of SIGGRAPH 2002*, July 2002.

[5] Robert D. Cook, David S. Malkus, and Michael E. Plesha. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, New York, third edition, 1989.

[6] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Proceedings of SIGGRAPH 2002*, pages 31–36, July 2002.

[7] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, July 1997.

[8] Eitan Grinspun, Petr Krysl, and Peter Schröder. Charms: A simple framework for adaptive simulation. In *Proceedings of SIGGRAPH 2002*, pages 281–290, July 2002.

[9] Doug L. James and Dinesh K. Pai. Artdefo - accurate real time deformable objects. In *Proceedings of ACM SIGGRAPH 99*, pages 65–72, August 1999.

[10] Doug L. James and Dinesh K. Pai. Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of SIGGRAPH 2002*, pages 582–585, July 2002.

[11] Zachi Karni and Craig Gotsman. 3D mesh compression using fixed spectral bases. In *Graphics Interface 2001*, pages 1–8, June 2001.

[12] Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of the ACM SIGGRAPH 2002 Symposium on Computer Animation*, pages 153–160, July 2002.

[13] Nuno M. M. Maia and Julio M. M. Silva, editors. *Theoretical and Experimental Modal Analysis*. Research Studies Press, Hertfordshire, England, 1998.

[14] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. In *Proceedings of ACM SIGGRAPH 92*, pages 309–312, July 1992.

[15] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proceedings of the ACM SIGGRAPH 2002 Symposium on Computer Animation*, pages 49–54, July 2002.

[16] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99*, pages 137–146, August 1999.

[17] James F. O'Brien, Chen Shen, and Christine M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *Proceedings of the ACM SIGGRAPH 2002 Symposium on Computer Animation*, pages 175–181, July 2002.

[18] Alex Pentland, Irfa Essa, Martin Friedmann, Bradley Horowitz, and Stan Sclaroff. The thingworld modeling system: Virtual sculpting by modal forces. In *1990 Symposium on Interactive 3D Graphics*, pages 143–144, March 1990.

[19] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. In *Proceedings of SIGGRAPH 89*, pages 215–222, July 1989.

[20] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface 95*, pages 147–154, May 1995.

[21] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of ACM SIGGRAPH 86*, pages 151–160, August 1986.

[22] Chen Shen, Kris K. Hauser, Christine M. Gatchalian, and James F. O'Brien. Modal analysis for real-time viscoelastic deformation. In *Proceedings of SIGGRAPH 2002*.

[23] Jos Stam. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum*, 16(3):159–164, August 1997.

[24] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[25] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–14, 1997.

[26] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. Foley automatic: Physically-based sound effects for interactive simulation and animation. In *Proceedings of SIGGRAPH 2001*, pages 537–544, August 2001.

[27] Kees van den Doel and Dinesh K. Pai. Synthesis of shape dependent sounds with physical modeling. In *Proceedings of the International Conference on Auditory Display (ICAD)*, 1996.

[28] Kees van den Doel and Dinesh K. Pai. The sounds of physical shapes. *Presence*, 7(4):382–395, 1998.

[29] Kesheng Wu and Horst Simon. TRLAN user guide. Technical Report LBNL-42953, Lawrence Berkeley National Laboratory, 1999.