Good Vibrations: Modal Dynamics for Graphics and Animation *

Alex Pentland and John Williams

Vision Sciences Group, E15-410
The Media Lab, M.I.T.
20 Ames St., Cambridge MA 02138

Abstract

Many of the problems of simulating and rendering complex systems of non-rigid objects can be minimized by describing the geometry and dynamics separately, using representations optimized for either one or the other, and then coupling these representations together. We describe a system which uses polynomial deformation mappings to couple a vibration-mode ("modal") representation of object dynamics together with volumetric models of object geometry. By use of such a hybrid representation we have been able to gain up to two orders of magnitude in efficiency, control temporal aliasing, and obtain simple, closed-form solutions to common (non-rigid) inverse dynamics problems. Further, this approach to dynamic simulation naturally lends itself to the emphasis and exaggeration techniques used in traditional animation.

1 INTRODUCTION

The idea of using computers to provide interactive simulation of non-rigid object dynamics has been a major goal of computer graphics, starting with Sketchpad [13], Thinglab [5], and the recent profusion of new computer graphics work on non-rigid dynamics [4,7,14]. Our project, which we have named Thingworld [10,11,12], was conceived as direct descendant of Sketchpad and Thinglab: our goal is to use interactive dynamic simulation of multibody situations to aid in physical design. In common with all previous attempts at achieving this goal, we have been confronted with the problem that the huge computational expense of calculating dynamic interactions prevents interactive simulation except for limited, toy situations. Furthermore, to be really useful to a designer, we must also be able to solve complex inverse dynamics problems, perform dynamic simulations for objects defined by large spline surfaces or by constructive solid geometry, and render objects without undue temporal aliasing — all of which are quite difficult with standard techniques and representations.

In the Thingworld system we have been able to minimize each of these problems by describing the geometry and dynamics separately, using representations optimized for either one or the other, and then coupling these representations together using deformation mappings. In our system dynamic properties are described by *modal analysis*, a method of breaking non-rigid dynamics down into the sum of independent vibration modes. The advantage of the modal approach is that it breaks the dynamics problem into many small, independent problems. This allows us to achieve a level of control not possible with the massed equations normally used in dynamic simulation. As a consequence many common inverse dynamics problems can be solved in closed form, and many traditional animation techniques can be easily automated.

Because formulations for describing non-rigid motion have been based on point-wise representations of shape, the detection and characterization of collisions has always been a major fraction of the computational cost in multibody simulation systems. Further, analytic models of geometry (e.g., β-splines) cannot be used because there has been no way to relate analytically-specified shape to object dynamics. Because the Thingworld system describes non-rigid deformation in terms of whole-body deformation modes, we can relate object dynamics to object shape via global polynomial deformation mappings. This allows us to couple non-rigid object dynamics with analytic models of geometry (in our case superquadrics) so that we can more efficiently and accurately characterize the forces produced by collisions.

The plan of this paper is to first present short description of the modal method for representing and calculating non-rigid object dynamics. We will then show how the modal representation can be modified to produce great gains in efficiency, to reduce temporal aliasing, and to solve inverse dynamics problems. We will then describe how the method can be generalized to arbitrary geometric representations, thus allowing more efficient and accurate detection and characterization of object collisions. Finally, we will discuss how this system can be adapted to automatically produce many of the effects used in traditional animation.

*This research was made possible by National Science Foundation Grant No. IRI-87-19920 and by ARO Grant No. DAAL03-87-K-0005

ACM-0-89791-312-4/89/007/0215 $00.75

2 MODAL DYNAMICS

2.1 Background: Finite Element Method

The finite element method (FEM) is a technique for simulating the dynamic behavior of an object. In the FEM the continuous variation of displacements throughout an object is replaced by a finite number of displacements at so-called nodal points. Displacements between nodal points are interpolated using a smooth function. Energy equations (or functionals) can then be derived in terms of the nodal unknowns and the resulting set of simultaneous equations can be iterated to solve for displacements as a function of impinging forces. In the dynamical case these equations may be written:

$$M\ddot{u} + D\dot{u} + Ku = f \qquad (1)$$

where u is a $3n$ x 1 vector of the (x, y, z) displacements of the n nodal points relative to the objects' center of mass, M, D and K are $3n$ by $3n$ matrices describing the mass, damping, and material stiffness between each point within the body, and f is a $3n$ x 1 vector describing the (x, y, z) components of the forces acting on the nodes. This equation can be interpreted as assigning a certain mass to each nodal point and a certain material stiffness between nodal points, with damping being accounted for by dashpots attached between the nodal points. The damping matrix D is often taken to be equal to sM for some scalar s; this is called mass damping.

To calculate the result of applying some force f to the object one discretizes the equations in time, picking an appropriately small time step, solves this equation for the new u, and iterates until the system stabilizes. Direct (implicit) solution of the dynamic equations requires inversion the K matrix, and is thus computationally expensive. Consequently explicit Euler methods (which are less stable, but require no matrix inversion) are quite often applied.

Even the explicit Euler methods are quite expensive, because the matrices M, D, and K are quite large: for instance, the *simplest* 3-D parabolic element produces 60 x 60 matrices, corresponding to the 60 unknowns in the 20 nodal points (x_i, y_i, z_i) which specify the element shape. In most situations M, D, and K are very much larger than 60 x 60, so that typically hundreds or thousands of very large matrix multiplications are required for each second of simulated time. For more details see references [4,7,14,15].

2.2 Modal Analysis

Because M, D and K are normally positive definite symmetric, and M and D are assumed to be related by a scalar transformation, Equation 1 can be transformed into $3n$ independent differential equations by use of the *whitening transform*, which simultaneously diagonalizes M, D, and K. The whitening transform is the solution to the following eigenvalue problem:

$$\lambda\phi = M^{-1}K\phi \qquad (2)$$

where λ and ϕ are the eigenvalues and eigenvectors of $M^{-1}K$.

Using the transformation $u = \phi\bar{u}$ we can re-write Equation 1 as follows:

$$\phi^T M\phi\ddot{\bar{u}} + \phi^T D\phi\dot{\bar{u}} + \phi^T K\phi\bar{u} = \phi^T f \quad . \qquad (3)$$

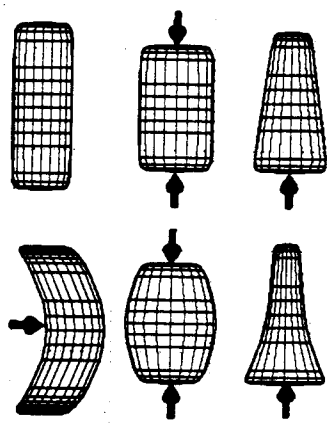

Figure 1: (a) A cylinder, (b) a linear deformation mode in response to compression, (c) a linear deformation mode in response to acceleration, (d) a quadratic mode in response to a bending force, (e) superposition of both linear and quadratic modes in response to compression, (f) superposition of both linear and quadratic modes in response to acceleration.

In this equation $\phi^T M\phi$, $\phi^T D\phi$, and $\phi^T K\phi$ are diagonal matrices, so that if we let $\bar{M} = \phi^T M\phi$, $\bar{D} = \phi^T D\phi$, $\bar{K} = \phi^T K\phi$, and $\bar{f} = \phi^T f$ then we can write Equation 3 as $3n$ independent equations:

$$\bar{M}_i\ddot{\bar{u}}_i + \bar{D}_i\dot{\bar{u}}_i + \bar{K}_i\bar{u}_i = \bar{f}_i \quad , \qquad (4)$$

where $\bar{M}_i$ is the i^{th} diagonal element of $\bar{M}$, and so forth. Because the modal representation diagonalizes these matrices it may be viewed as *preconditioning* the mass and stiffness matrices, with the attendant advantages of better convergence and numerical accuracy.

What Equation 4 describes is the time course of one of the object's *vibration modes*, hence the name *modal analysis* [16]. The constant $\bar{M}_i$ is the generalized mass of mode i, that is, it describes the inertia of this vibration mode. Similarly, $\bar{D}_i$, and $\bar{K}_i$ describe the damping and spring stiffness associated with mode i, and $\bar{f}_i$ is the amount of force coupled with this vibration mode. The i^{th} row of ϕ describes the *deformation* the object experiences as a consequence of the force $\bar{f}_i$, and the eigenvalue λ_i is proportional to the natural resonance frequency of that vibration mode.

Figure 1 illustrates the some of the first and second order modes of a cylinder. Figure 1(a) shows the cylinder at rest, (b) shows the cylinder experiencing a linear deformation in response to a compressive force, (c) shows the cylinder experiencing a linear shear deformation in response to an accelerating force, (d) shows a quadratic deformation in response to a centrally-applied (bending) force, and (e) and (f) show how both the linear and second order deformations can be superimposed to produce a more accurate simulation of the object's response to the compressive and accelerating forces shown in (b) and (c).

To obtain an accurate simulation of the dynamics of an object one simply uses linear superposition of these modes to determine how the object responds to a given force. Be-

cause Equation 4 can be solved in closed form, we have the result that for objects composed of linearly-deforming materials *the non-rigid behavior of the object in response to an impulse force can be solved in closed form for any time t.* The solution is discussed in Section 5.1. In environments with more complex forces, however, analytic solution becomes cumbersome and so numerical solution is preferred. Either explicit or implicit solution techniques may be used to calculate how each mode varies with time.

Non-linear materials may be modeled by summing the modes at the end of each time step to form the material *stress state* which can then be used to drive nonlinear plastic or viscous material behavior.

3 USING THE MODAL METHOD

Although the simple modal method offers some benefits in terms of efficiency and stability, its main advantage is that it allows us to control the computation in ways that are advantageous to various applications. In this section we detail some of the variations on the basic modal method that we have found to be particularly useful.

3.1 Increased Speed

Modes associated with high resonance frequencies normally have little effect on object shape. This is because, for a fixed excitation energy, the displacement amplitude for each mode is inversely proportional to the *square* of the mode's resonance frequency. Thus a relatively accurate and more efficient simulation of an object's dynamics can be accomplished by discarding the small-amplitude, high-frequency modes, and superimposing only the large-amplitude, low-frequency modes. We can determine which modes to discard by examining their associated eigenvalue, which is proportional to the resonance frequency.

The amount of error introduced by discarding high-frequency modes can be checked by occasionally substituting the displacements $\mathbf{u}$ produced by low-frequency modal analysis into the full equations. When significant error is found additional modes can be added. Exactly which modes to add can be determined by a principal components analysis of the error residuals.

One effect of discarding modes is, of course, to reduce the number of equations that must be considered within each time step. However, because the maximum allowable time step is inversely proportional to the highest resonance frequency in the system of equations, a more important effect of discarding high-frequency modes is that we can use much larger time steps. In typical situations we have found that the savings from fewer equations and larger times steps can reduce computation time by up to two orders of magnitude, while at the same time producing a reasonably accurate, realistic-looking animation.

3.1.1 Number of modes required

For the sake of increased efficiency, our approach has been to model only as many modes as are required. In a quick-and-dirty analysis — often sufficient during the initial phase of a design — only rigid-body or rigid-body plus linear strain modes may be used, resulting in large computational savings. Later, more modes can be added to achieve any level of desired accuracy, although at greater cost.

We have found that most commonplace multi-body interactions can be adequately modeled by use of only rigid-body, linear, and quadratic strain modes, as is shown in Figures 1 and 2. Note that this is *not* true for bodies whose dimensions are quite disparate, however it is exactly these cases that can be adequately treated by either a one or two dimensional analysis, and thus are cases where the standard FEM is quite efficient.

3.1.2 Recomputing matrices and modes

Normally, in either the finite element or modal methods, the mass, damping, and stiffness matrices are not recomputed at each time step. The use of fixed M, D, and K (or, equivalently, fixed modes) is well-justified as long as the material displacements are small. The definition of "small," however, is quite different for different modes. Because the eigenvalue decomposition in Equation 2 performs a sort of principal-components analysis, it is the gross object shape (e.g., its low-order moments of inertia) determine the low-frequency modes, which as a consequence are quite stable. High-frequency modes are much less stable because they are determined by the fine features of the object's shape.

In the standard finite element formulation the action of each mode is distributed across the entire set of equations, so that one must recompute the mass and stiffness matrices as often as required by the very highest-frequency vibration modes. When these high-frequency modes are discarded the mass, damping, and stiffness matrices need to be recomputed much less frequently — a large computational savings. We have found, for instance, that in most animation sequences we can use a single, fixed set of low-frequency modes throughout the entire simulation.

3.1.3 An example

Figure 2 shows a example of computing non-rigid dynamic interaction: a ball colliding with a two-by-four. As can be seen, the interaction and resulting deformations look realistic despite the use of only first and second order modes. Perhaps the most impressive fact about this example, however, is the speed of computation: Using a Symbolics 3600 (with a speed of roughly one MIP), it requires only one CPU second to compute each second of simulated time!

3.2 Temporal Aliasing

One important side effect of discarding high-frequency modes is reduction in temporal aliasing artifacts. A dynamic simulation using the standard finite element method will produce very many small, high-temporal-frequency displacements. This is especially true for stiff materials. To avoid temporal aliasing artifacts these small displacements must be accurately tracked (requiring a small time step), and then averaged over time to produce each image. In modal analysis these high frequency displacements can be directly identified and discarded, thus reducing not only the number of time steps required, but also the need for time averaging in order to avoid temporal artifacts.

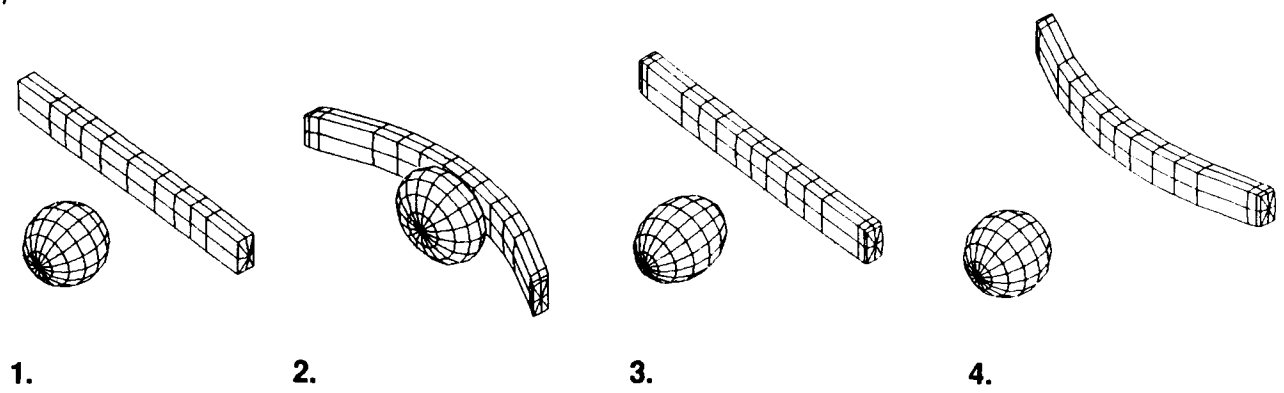

Figure 2: A ball colliding with a two-by-four

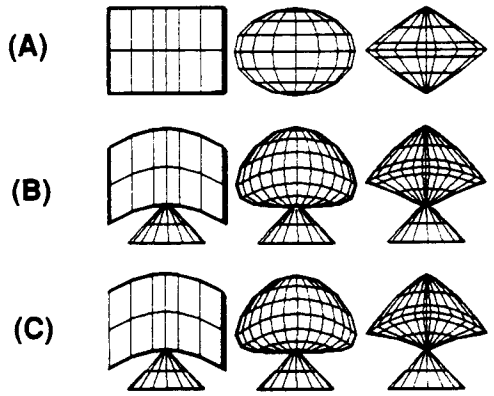

Figure 3: Low-order deformation modes are visually similar when objects have similar low-order moments of inertia.

3.3 Use of Approximate Modes

An object's low-order deformation modes can be thought of as the principal components of the object's repertoire of non-rigid behavior. These modes can be found by solving the eigenvalue problem of Equation 2, however for visualization purposes we have found that it is sufficient to use fixed, precomputed deformation modes that are parameterized only by the object's low order moments of inertia.

This is illustrated in Figure 3, which shows three objects colliding with a post after having been dropped from a few feet above the post. Figure 3(a) shows the original, undeformed objects. Figure 3(b) shows the collisions simulated using modes computed by solving Equation 2. In Figure 3(c) we have precomputed the modes of a rectangular solid with approximately the same moments as the object to be animated. These precomputed deformation modes were then used in place of the object's true modes in making the animation. Despite use of precomputed modes it can be seen that the collisions are visually very similar.

A more accurate variation on this theme is to use 20 nodal points (i.e., a simple 3-D parabolic element) to approximate the shape of the object to be animated. Equation 2 can be solved relatively quickly for this number of nodal points, and the resulting modes will produce a reasonably accurate simulation. Such shortcuts to finding an object's modes can produce an important savings in interactive simulation systems such as Thingworld, where the user frequently changes each object's static geometry.

4 COMBINING DYNAMICS AND ANALYTIC GEOMETRY

One problem with standard non-rigid dynamical techniques is that they are based on use of a point-wise representation of geometry, thus forcing the representation of geometry and dynamics to be identical. As a consequence one cannot, for instance, specify details of geometry without incurring large costs in calculating dynamic behavior, nor can one directly animate objects defined by, for example, large spline patches or constructive solid geometry. The fact that the same representation must be used for both geometry and dynamics thus has a large impact upon the efficiency and accuracy of multibody simulations, where detailed specification of geometry is required to obtain accurate detection and characterization of collisions.

We have been able to combine separate representations of dynamic behavior and geometric form in order to avoid these problems. We have accomplished this by describing each mode by an appropriate polynomial function, and then using global deformation techniques [3] to establish the correspondence between dynamic state and geometric state. The result is an efficient scheme for simulating non-rigid dynamics that can be applied in a unified manner to objects whose geometry is defined using a wide range of techniques.

To accomplish this, we must first realize that modes may be classified by the complexity of the associated deformation, e.g., as 0^{th} order (rigid body) modes, 1^{st} order (linear deformation) modes, 2^{nd} order (quadratic deformation) modes, and so forth, as was illustrated by Figure 1. Thus we can describe the deformation associated with each mode by use of polynomial deformation mappings of the appropriate degree. This is accomplished by performing a linear regression of a polynomial with m terms in appropriate powers of x, y, and z, against the n triples of x, y and z coefficients that compose ϕ_i, a $3n$ x 1 vector containing the elements of the

i^{th} row of ϕ:

$$\alpha = (\beta^T\beta)^{-1}\beta^T\phi_i \quad , \tag{5}$$

where α is an m x 1 matrix of the coefficients of the desired deformation polynomial, β is an $3n$ x m matrix whose first column contains the elements of $\mathbf{u} = (x_1, y_1, z_1, x_2, y_2, z_2, ...)$, and whose remaining columns consist of the modified versions of u where the x, y, and/or z components have been raised to the various powers, e.g.,

$$\beta = \begin{pmatrix} x_1 & x_1^2 & x_1 & x_1 & \cdots \\ y_1 & y_1 & y_1^2 & y_1 & \cdots \\ z_1 & z_1 & z_1 & z_1^2 & \cdots \\ x_2 & x_2^2 & x_2 & x_2 & \cdots \\ y_2 & y_2 & y_2^2 & y_2 & \cdots \\ z_2 & z_2 & z_2 & z_2^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{pmatrix} . \tag{6}$$

The question of which polynomial powers are the appropriate for a particular column of ϕ can be decided either by inspection (noting that the order of the deformation is related to the associated eigenvalue), or automatically by including all combinations of powers of x, y, and z (up to some limit), performing the regression, and then discarding coefficients with negligible magnitude.

The result is a polynomial model of the unit amplitude deformation associated with mode i. By simply scaling this polynomial deformation according to the mode's amplitude we can accurately copy the effects of this mode on the object's shape. By superimposing these deformations we obtain an accurate accounting of the object's non-rigid deformation.

4.1 Fast Collision Characterization

In complex, multi-body simulations the ability to efficiently detect and characterize collisions is extremely important. Unfortunately, the point-wise representations used by the standard FEM are quite poor at this task. When using a polygon representation, for instance, the computational complexity of collision detection is $O(nm)$ operations, where n is the number of polygons and m is the number of points to be considered after pruning via bounding box calculations [9].

In contrast, one can perform collision detection relatively efficiently when employing volumetric representations (e.g., superquadrics [2,6,10]) by making use of their inside-outside function. In our system the basic volumetric primitive is a superquadric, which is mapped from its canonical reference frame[1] to its three-space position by an affine transformation T. The normalized inside-outside function $D(x, y, z)$ for superquadrics is:

$$D(x,y,z) = \left[((x/a_1)^{2/\epsilon_1} + (y/a_2)^{2/\epsilon_1})^{\epsilon_1/\epsilon_2} + (z/a_3)^{2/\epsilon_2}\right]^{\epsilon_2} \tag{7}$$

where the position (x, y, z) is relative to the object's canonical reference frame. The basic operation for collision detection, then, is to take points (x, y, z) sampled from the tested object's surface, apply T^{-1} to convert them to the canonical reference frame, and then substitute them into the inside-outside function. When the result is less than one the point is inside the surface, if greater then one the point is outside. Thus, the computational complexity is only $O(m)$, rather than $0(nm)$, where n and m are as before. As with other representations [9], to find the exact point in space-time at which contact between the two bodies occurred requires use of numerical minimization techniques, where both point position, T and Equation 7 are expressed as functions of time. In the Thingworld system we have found that the ability to perform fast collision detection using volumetric representations yields large computational savings.

[1]The canonical reference frame is when the object has zero rotation, and is centered at $(0, 0, 0)$

4.2 Accuracy of Collision Characterization

A more subtle but perhaps equally important advantage of being able to use analytic representations in dynamic simulations is the ability to characterize the collision surface more quickly and precisely. For instance, one difficult problem that arises when using any discrete time technique is that colliding bodies often interpenetrate during a time step. The depth, area and shape of this penetration determines the repulsive force generated.

With point-wise (polygon) representations it is difficult to determine the interpenetration region, so that most systems ignore the contact area's shape and simply find the single point (normally a polygon vertex) that first contacted the surface. As a consequence the calculated force is often seriously in error. When using analytic representations of geometry, however, both surface normal and principal curvatures are readily available so that good closed form approximations to the depth, area and shape of the interpenetration region can be easily computed.

5 The Right Control Knobs

One of the most important aspects of any simulation or animation system is the ability to control the behavior of objects in a natural, intuitive, and convenient manner: in short, the system must have the right control knobs. In simulation systems such as Thingworld, one often needs to be able to solve simple inverse dynamics problems: For instance, to make something jump from here to there and land softly. In animation systems the same requirements arise, but in addition one needs to be able to produce pleasing but non-physically-realistic effects. In traditional animation some of the most important of these effects are called squash-and-stretch, anticipation, and exaggeration [8].

The control knobs for these sorts of things simply don't exist with standard approaches to dynamic simulation. Even simple inverse dynamics problems, for instance, require solving huge numerical minimization problems because all of dynamical equations are closely coupled together. Similarly, traditional animation effects such as squash-and-stretch can only be obtained by carefully jiggering material properties and external forces as a function of object position, velocity, and so forth.

The situation is quite different when using modal analysis, because closed-form solutions exist for each mode's behavior as a function of time, and because the various modal behaviors are independent of each other so that they may

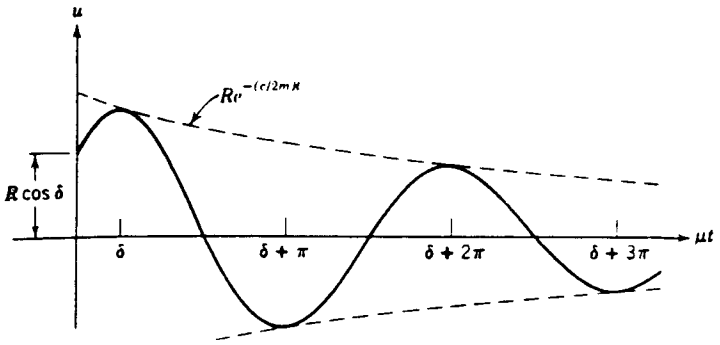

Figure 4: Damped vibration as a function of time.

be treated separately. Further, the low-order modes of an object seem to closely mimic many of the effects used in traditional animation.

5.1 Inverse Dynamics

The time behavior of each mode in response to an impinging force is given by Equation 4. The generic solution to this equation is

$$\begin{aligned}
\bar{u}_i &= Ae^{r_1 t} + Be^{r_2 t}, \\
&\quad \text{for} \quad \bar{D}_i^2 - 4\bar{K}_i\bar{M}_i > 0, \quad r_1, r_2 < 0 \\
\bar{u}_i &= (A + Bt)e^{(\bar{D}_i/2\bar{M}_i)t}, \\
&\quad \text{for} \quad \bar{D}_i^2 - 4\bar{K}_i\bar{M}_i = 0, \\
\bar{u}_i &= e^{(\bar{D}_i/2\bar{M}_i)t}(A\cos\mu t + B\sin\mu t), \\
&\quad \text{for} \quad \mu = (4\bar{K}_i\bar{M}_i - \bar{D}_i^2)^{1/2}/2\bar{M}_i > 0
\end{aligned} \tag{8}$$

for the overdamped, critically damped, and underdamped cases, where

$$r_1, r_2 = \frac{-\bar{D}_i \pm \sqrt{\bar{D}_i^2 - 4\bar{K}_i\bar{M}_i}}{2\bar{M}_i} \tag{9}$$

and A and B depend on the initial conditions [1]. The third case, underdamped motion, occurs most commonly in mechanical systems and is referred to as "damped vibration." To see this we let $A = R\cos\delta$ and $B = R\sin\delta$ in Equation 8 to obtain

$$\bar{u}_i = Re^{-(\bar{K}_i/2\bar{M}_i)t}\cos(\mu t - \delta) \quad , \tag{10}$$

which is graphed in Figure 4.

Thus, once we know the amplitude and derivative of a mode at time zero, we can predict its behavior for all future times — or at least until an external force adds or subtracts energy from the mode. In particular, given initial conditions $\bar{u}_i(0) = \chi$, $\dot{\bar{u}}_i(0) = \dot{\chi}$, and underdamped free oscillation, then

$$\bar{u}_i(t) = \left(\chi^2 + (\frac{\dot{\chi}}{\mu} + \frac{\bar{D}_i\chi}{2\bar{M}_i\mu})^2\right)^{1/2} e^{\bar{K}_i t/2\bar{M}_i}\cos(\mu t - \delta) \tag{11}$$

where $\delta = \tan^{-1}[(\dot{\chi}/\chi + \bar{D}_i\chi/2\mu\bar{M}_i)/\chi]$. Using this relation we can achieve a desired object shape at some time t_1 by adjusting initial modal amplitude and velocity at time t_0. Similarly, we can specify the desired object shape at times t_0 and t_1, and then solve for the force required to achieve those

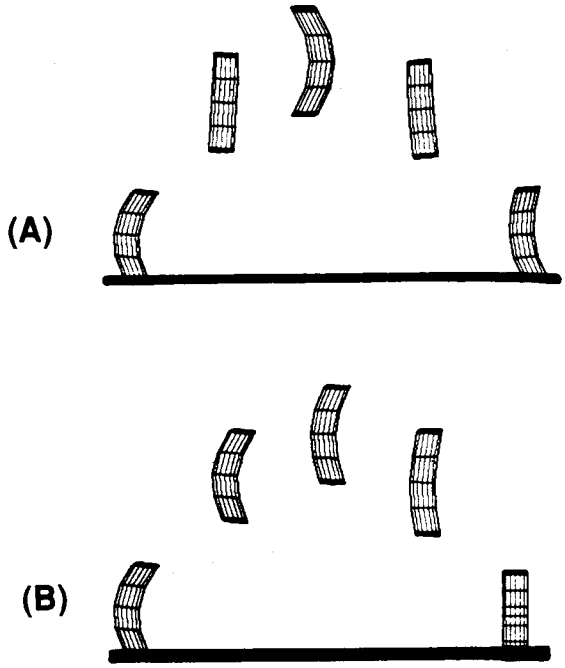

Figure 5: A time lapse illustration of a cylinder jumping and landing (a) with a hard thump, (b) softly. Time proceeds from left to right.

constraints. Thus, Equation 11 can provide us with closed-form solutions to many common inverse dynamics problems. For closed-form solutions under other initial conditions see reference [1].

5.1.1 Some Examples

Imagine that we want a cylindrical solid to jump from point A to point B, landing either softly or with a hard "thump". Further, imagine that we wish to control the cylinder by changing only the characteristics of it's "muscles," i.e., by controlling the displacement and spring constant associated with each deformation mode. The inverse dynamics problem, then, is to make sure that the cylinder has the correct amount of extension or compression at the point of landing so that it can achieve the desired type of landing.

The mathematics for calculating a trajectory that will take the cylinder from point A at time $t = t_0$ to point B is well known. That calculation will also give us the time $t = t_1$ at which landing will occur, and the force vector $f_{initial}$ needed to achieve the jump. If we idealize the geometry and time course of how the cylinder pushes against point A, then we can use standard kinematics to determine how much the cylinder must "crouch" and tense it's "muscles" (i.e., what initial modal displacements $\bar{u}_i(t_0)$ and spring constants $\bar{K}_i(t_0)$ are required) in order to produce the desired force vector.

Producing a jump by use of the spring energies stored in the various modes will leave each of the modes in some state $\bar{u}_i(t_0 + \epsilon) = \chi_i$, $\dot{\bar{u}}_i(t_0 + \epsilon) = \dot{\chi}_i$ as the cylinder leaves the surface. The inverse dynamics problem is then to set the spring constants $\bar{K}_i(t)$ (for $t_0 < t < t_1$) of the cylinder's

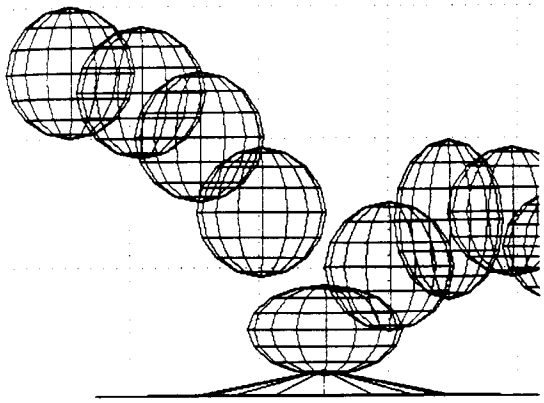

Figure 6: (a) Physical squash-and-stretch in a collision.

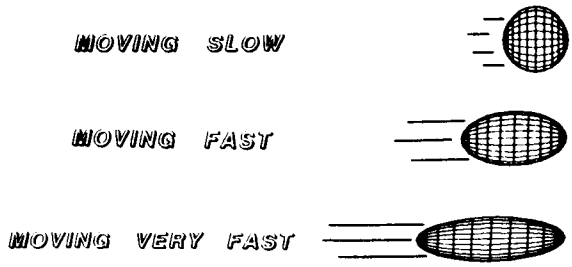

Figure 7: Stretching/squashing tied to velocity

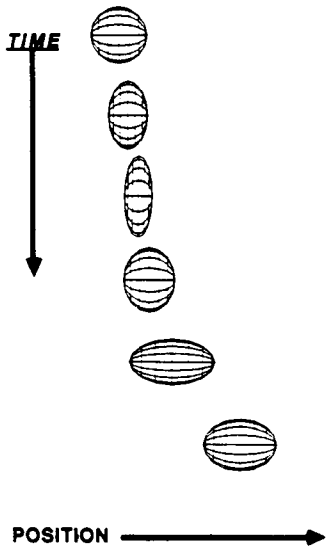

Figure 8: Stretching/squashing tied to speed minus acceleration; time proceeds from top to bottom.

"muscles" so that the natural oscillation of the cylinder exerts the desired motion-canceling force f_{final} on point B at the appointed instant in time.

To obtain f_{final} at $t = t_1$ we first determine what combination of modal amplitudes will exert the desired final force by computing

$$\bar{u}_i(t_1) = \bar{f}_i / \bar{K}_i(t_1) \quad , \tag{12}$$

where $\bar{f} = \phi^T f_{final}$, and the $\bar{K}_i(t_1)$ correspond to "tensing" the muscles for landing. We then solve Equation 11 with the given values of $t = t_1$, $\bar{M}_i$, $\bar{D}_i$, $\bar{u}_i(t_0) = \chi_i$, and $\dot{\bar{u}}_i(t_0) = \dot{\chi}_i$ in order to find a stiffness $\bar{K}_i(t)$, $t_0 < t < t_1$, such that $\bar{u}_i(t_1)$ has the desired value. [2] Thus Equation 11 provides a closed-form solution to such simple inverse dynamics problems — at least when we can idealize contact geometry, friction, etc. Examples of jumps computed in this manner are shown in Figure 5.

In most situations of interest, unfortunately, the particulars of geometry and friction are sufficiently complex and non-linear that there is no closed-form solution, so that one must still employ the sort of constrained minimization described in [17] to obtain a solution. However, as these examples illustrate, by using Equation 11 it appears that the problem can be reduced from several thousand free parameters to only a few dozen free parameters.

5.2 Control of Animation

The deformations caused by an object's low-order vibration modes correspond closely to the types of exaggeration and emphasis used in traditional animation. Thus, the amplitude of these low-order modes provides the control knobs needed for such animation.

A simple version of squash-and-stretch in collisions is well modeled by straightforward application of non-rigid dynamic simulation: things *do* squash and stretch during collisions, a shown in Figure 6. However, as applied in traditional animation, this notion goes well beyond simply obtaining physically-realistic deformations during a collision. It also occurs as a response to motion, to acceleration, and even in response to emotional states. By providing a "stretch/squash" control knob we can wire squashing-type deformations directly to other parameters, both physical and non-physical, in order to obtain interesting visual effects.

Examples of this are shown in Figures 7, 8 and 9, where we have wired the "stretch/squash" knob to various physical properties. Figure 7 shows three objects moving at different speeds. In this figure the amplitude of the stretching/squashing deformation is set equal the speed, so that as the object moves faster it becomes stretched out in the direction of motion.

Figure 8 shows a time series where the stretch/squash deformation is equal to the speed minus the acceleration, so that an accelerating object "piles up" in anticipation as the motion begins, and stretches out as the motion reaches steady state.

Finally, Figure 9 shows three mushrooms with both bending and stretching/squashing deformations tied to image x position: as a consequence the mushrooms "wilt" from left to right. The same deformations could be tied to emotional state, for instance, thus providing physical illustration of a character's state of depression or elation.

[2] The $\bar{K}_i(t_1)$ must of course be large enough that the modal displacements at $t = t_1$ (which are no larger than the displacements at $t = t_0$) generate sufficient energy.

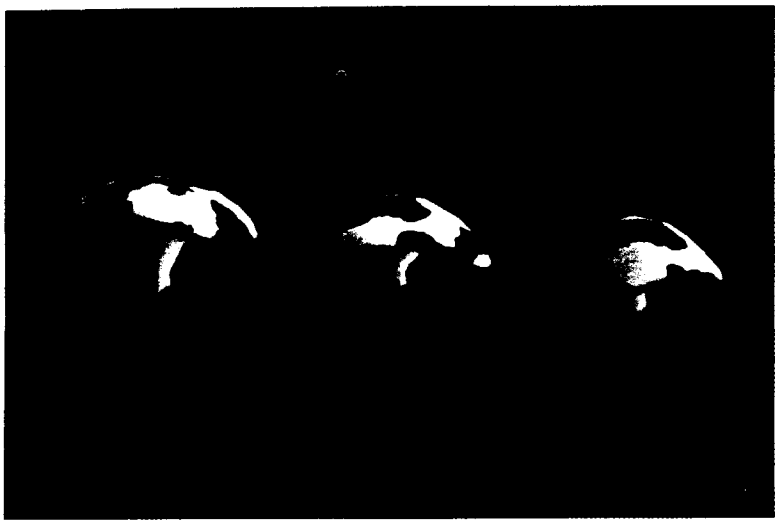

Figure 9: Stretching/squashing to express emotional state.

6 SUMMARY

The idea of using computers to provide interactive simulation of non-rigid object dynamics has long been frustrated by the inability to efficiently calculate dynamic interactions, to solve inverse dynamics problems, to use geometry defined by splines or constructive solid geometry, and to avoid temporal aliasing problems. We have been able to minimize each of these problems by developing new, hybrid methods for representing and calculating object dynamics in which the object's geometry and dynamics are described by separate but yoked representations. The result is a system which is efficient at performing dynamic simulations, can be applied to a wide range of geometric models, and which is useful for implementing many of the techniques used in traditional animation.

REFERENCES

[1] Anderson, J. S., and Bratos-Anderson, M., (1987) Solving Problems in Vibrations, Longman Scientific and Technical Publ., Essex, England.

[2] Barr, A., (1981) Superquadrics and angle-preserving transformations, *IEEE Computer Graphics and Application, 1* 1-20

[3] Barr, A., (1984) Global and local deformations of solid primitives. Proceedings of SIGGRAPH '84, *Computer Graphics 18*, 3, 21-30

[4] Barzel, R., and Barr, A., (1988) A Modeling System Based On Dynamic Constraints, Proceedings of SIGGRAPH '88, *Computer Graphics*, Vol. 22, No. 4, pp. 179-188

[5] Borning, A., (1979), Thinglab – a constraint-oriented simulation laboratory. SSL-79-3, Xerox PARC, Palo Alto, CA.

[6] Gardiner, M. (1965) The superellipse: a curve that lies between the ellipse and the rectangle, *Scientific American*, September 1965.

[7] Issacs, P., and Cohen, M., (1987) Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions, and Inverse Dynamics, Proceedings of SIGGRAPH '87, *Computer Graphics*, Vol. 21, No. 4, pp 215-224.

[8] Lasseter, J., (1987) Principles of Traditional Animation Applied to 3D Computer Animation, Proceedings of SIGGRAPH '87, *Computer Graphics 21*, 4, 35-44

[9] Moore, M., and Wilhelms, J., (1988) Collision Detection and Response for Computer Animation, Proceedings of SIGGRAPH '88, *Computer Graphics 22*, 4, 289-298

[10] Pentland, A. (1986) Perceptual Organization and the Representation of Natural Form, *Artificial Intelligence Journal*, Vol. 28, No. 2, pp. 1-38.

[11] Pentland, A., (1987) Towards and Ideal 3-D CAD System, *SPIE conference on Machine Vision and the Man-Machine Interface*, Jan. 11-16, San Diego, CA.

[12] Pentland, A., and Williams, J. (1988) Virtual Construction, *Construction*, Vol. 3, No. 3, pp. 12-22

[13] Sutherland, I., (1963), Sketchpad: A Man-Machine Graphical Communications System, in Interactive Computer Graphics, in *1963 Spring Joint Computer Conference*, reprinted in H. Freeman, ed., IEEE Comp. Soc., 1980, pp. 1-19.

[14] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K., (1987) Elastically deformable models, Proceedings of SIGGRAPH '87, *Computer Graphics*, Vol. 21, No. 4, pp 205-214.

[15] Williams, J., Musto, G., and Hawking, G., (1987) The Theoretical Basis of the Discrete Element Method, *Numerical Methods in Engineering, Theory, and Application*, Rotterdam: Balkema Publishers

[16] Williams, J. and Musto, G. (1987) Modal Methods for the Analysis of Discrete Systems, *Computers and Geotechnics*, Vol. 4, pp 1-19.

[17] Witkin, A., and Kass, M. (1988) Space-Time Constraints, Proceedings of SIGGRAPH '88, *Computer Graphics 22*, 4, 159-168