Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation

Min Gyu Choi and Hyeong-Seok Ko

Abstract— **This paper proposes a real-time simulation technique for large deformations. Green's nonlinear strain tensor accurately models large deformations; however, time stepping of the resulting nonlinear system can be computationally expensive. Modal analysis based on a linear strain tensor has been shown to be suitable for real-time simulation, but is accurate only for moderately small deformations. In the present work, we identify the rotational component of an infinitesimal deformation, and extend traditional linear modal analysis to track that component. We then develop a procedure to integrate the small rotations occurring at the nodal points. An interesting feature of our formulation is that it can implement both position and orientation constraints in a straightforward manner. These constraints can be used to interactively manipulate the shape of a deformable solid by dragging/twisting a set of nodes. Experiments show that the proposed technique runs in real-time even for a complex model, and that it can simulate large bending and/or twisting deformations with acceptable realism.**

Index Terms— **Physically Based Modeling, Physically Based Animation, Deformation, Modal Analysis**

I. Introduction

EVERYTHING in this world deforms. In many objects or creatures, deformation is such a conspicuous characteristic that their synthetic versions look quite unnatural if the deformation process is not properly simulated. Therefore, modeling of deformation is an important aspect of computer animation production. This paper presents a physically-based technique for dynamic simulation of deformable solids, attached to rigid supports and excited by their rigid motions and/or external forces such as gravity. The proposed technique makes a significant improvement in simulation speed, while maintaining the realism to a sufficient level, even for large deformations.

It is a well-established approach to model elastic solids as continuums and solve their governing equations numerically using finite element methods. When adopting a continuum model, it is necessary to choose the measure of strain that quantifies deformation. Green's strain tensor, which consists of linear terms and a nonlinear term, has been a common choice for large deformations. Unfortunately, time stepping of the resulting nonlinear system can be computationally expensive, hampering its practical use in animation production.

The computational load can be reduced remarkably by employing *modal analysis* [19] based on a linear strain tensor. In this technique, a set of deformation modes – a small number of principal shapes that can span free vibration of the elastic model – is identified and precomputed. Then, the problem of simulating deformation is transformed to that of finding the weights of the modes, which results in a significant reduction in computational complexity. This technique can also synthesize geometrically complex deformations with negligible main CPU costs on programmable graphics hardware [10].

However, modal analysis can produce quite unnatural results when applied to bending or twisting deformations of relatively large magnitudes. In particular, the volume of the deformed shape can increase unrealistically, as shown in Fig. 4. These unnatural results are due to the omission of the nonlinear term, which is not negligible for such deformations. In this paper, we propose a new technique that overcomes the above limitations of linear modal analysis. As a result, the proposed technique generates visually plausible shapes of elastodynamic solids undergoing large rotational deformations, while retaining its computational stability and speed. Also, our formulation provides a new capability for orientation constraints, which has not been addressed in previous studies. The use of position/orientation constraints can create interesting animations (Section VI) which would have been difficult if orientation constraints were not provided.

The innovative aspect of our technique lies in the way of handling rotational parts of deformation in the modal analysis framework. To exploit the framework of linear modal analysis, we omit the nonlinear term during the initial setup, which corresponds to precomputing the modal vibration modes at the rest state. When the simulation is run, however, we keep track of the local

The authors are at Graphics & Media Lab., School of Electrical Engineering and Computer Science, Seoul National University, 151-742 Gwanak-gu, Seoul, Korea. http://graphics.snu.ac.kr/{∼mgchoi|∼ko}

rotations that occur during the deformation, based on the infinitesimal rotation tensor. Then, at each time step we warp the precomputed modal basis in accordance with the local rotations of the mesh nodes. The rest of the method is basically the same as in linear modal analysis. The above book-keeping operations – tracking local rotations and warping the modal basis – require only a small amount of extra computation. Therefore, as in [10], our method can simulate dynamic deformations in real-time by employing programmable graphics hardware, but with an extended coverage of deformations.

II. Related Work

Since the pioneering work of Terzopoulos et al. [24], much effort has been devoted to simulating the motion of deformable objects. Past studies in this area have had two central aims: to speed up the simulation and/or increase the realism of the result. A comprehensive survey of this subject can be found in computer graphics literature [6] and mechanics literature [1], [26].

The speed and realism of simulations, which usually trade off each other, are heavily dependent on how the nonlinearities are handled. If realism is important, Green's quadratic strain tensor could be used, which produces realistic results even for large deformations. However, time stepping of the resulting nonlinear system can be computationally expensive. Several methods have been proposed to reduce the computational load of this approach. Lumped mass approximation diagonalizes the mass matrix so that its inverse can be computed efficiently. Further reduction of the computation time can be achieved by employing adaptive methods based on a multi-grid solver [23], non-nested overlapping layers of unstructured meshes [3], subdivision of the control lattice [2], or refinement of basis functions [7]. However, the speed-up achieved by those methods is limited, because they must still deal with the inherent problems resulting from the nonlinearities.

The computation time can be greatly reduced by adopting the modal analysis of linear elastodynamics, which omits the nonlinear term. Since Pentland and Williams [19] first introduced this technique to the computer graphics community, it has been used for modeling the dynamic movements of trees in turbulent wind [22], and for generating sounds corresponding to the behavior of deformable objects [18]. In particular, James and Pai [10] showed that the deformation of human skin excited by rigid body motion can be generated in real-time on programmable graphics hardware. They also proposed an output-sensitive technique for collision detections among reduced deformable models [11]. Hauser et al. [8] addressed the manipulation constraints, and combined modal analysis with rigid body simulation to deal with free-floating deformable objects. Although modal analysis significantly accelerates the simulation, it generates noticeable artifacts when applied to large deformations due to the linearization. Here we propose a technique that eliminates the linearization artifacts while retaining the efficiency of the modal analysis.

The linearization artifacts observed in simulations based on linear modal analysis arise in large part because linear modal analysis does not account for rotational deformations. Terzopoulos and Witkin [25] introduced a frame of reference and modeled the deformation relative to that reference frame. Since simulations using the reference frame capture the rotational part of the deformation, they can handle large rotational motions of deformable solids. However, large deformations within the solid are also susceptible to the linearization artifact. To realistically animate articulated deformable characters, Capell et al. [2] developed a method in which the character is first divided into overlapping regions, then each region is simulated separately, and finally the results are blended. For nonlinear quasi-static deformations of articulated characters, Kry et al. [12] introduced a modal displacement model equipped with a continuously articulated coordinate system.

To address large relative rotational deformations within a single object, Müller et al. [14] proposed the stiffness warping method that tracks the rotation of each node and warps the stiffness matrix. Our method is similar to their approach in that rotations are handled separately to reduce the linearization artifacts. The intrinsic difference is that, whereas the stiffness warping method is formulated in the original space, our method is formulated in the modal space. This results in a significant speed up in both simulation and visualization by (a) solving decoupled, reduced system of linear equations, and (b) utilizing programmable graphics hardware for vertex updates of large models. However, unlike the corotational methods [4], [5], [15] that employ element-wise rotation, both Müller et al.'s work [14] and our work are based on node-wise rotation of the stiffness matrix, thus can produce a spurious ghost force when applied to a free-floating deformable object. Currently, our work is focused on a deformable object attached to a rigid support, thus the ghost force effects are suppressed by the constraint force.

Recently, James and Fatahalian [9] proposed data-driven tabulation of the state space dynamics and dimensional model reduction of the deformed shapes to simulate large deformations at an interactive speed with visually realistic results. Because the tabulation could not be performed for all possible system responses, they

confined user interactions to certain types of movements. They reported that the precomputation for the dinosaur model shown in Fig. 10 took about 30 hours. In comparison, our method is formulated by adding simple extensions to linear modal analysis. As a consequence, it does not entail long precomputation times, nor does it restrict the types of user interactions. However, self-collisions and global scene illumination cannot be precomputed in our method, which was possible in [9].

III. Rotational Part in a Small Deformation

The nonlinear term in the strain tensor is responsible for the appearance and disappearance of rotational deformations. However, because the strain tensor used in the present work does not include the nonlinear term, a straightforward modal analysis will not generate such phenomena, and will therefore give rise to visual artifacts for large deformations.

Even though a linear strain tensor does not properly model the rotational deformation, fortunately, investigating the kinematics of deformation provides a clue to lessen such an inability; In fact, it has been generally known that every infinitesimal deformation can be decomposed into a rotation followed by a strain [21]. This finding forms the basis of the technique proposed here. Specifically, at every time step of the deformation simulation, we first identify the (small) rotations occurring over the material points, and then integrate the effects of those rotations to obtain the deformed shape.

This section commences with an investigation of the kinematics of infinitesimal deformation to show how such deformations can be decomposed into a strain and a rotation. This analysis is entirely based on the mechanics literature [21]. We then show how this decomposition can be used to extend modal analysis so that it keeps track of rotations, while still retaining the basic framework of modal analysis. The method for integrating the effects of rotations will be presented in the next section.

A. Kinematics of Infinitesimal Deformation

Before introducing the decomposition of infinitesimal deformations, we first define the necessary notations. Suppose that $\mathbf{x} \in \mathbb{R}^3$ denotes the position of a material point of an elastic solid in the undeformed state, which moves to a new position $\mathbf{a}(\mathbf{x})$ due to a subsequent deformation. To focus on the displacements caused by the deformation, we make use of the *displacement field*, $\mathbf{u} : \mathbb{R}^3 \to \mathbb{R}^3$, such that

$$\mathbf{a}(\mathbf{x}) = \mathbf{x} + \mathbf{u}(\mathbf{x}), \quad \mathbf{x} \in \Omega,$$

where Ω is the domain of the solid. Then, differentiating both sides of the above equation with respect to $\mathbf{x}$ gives

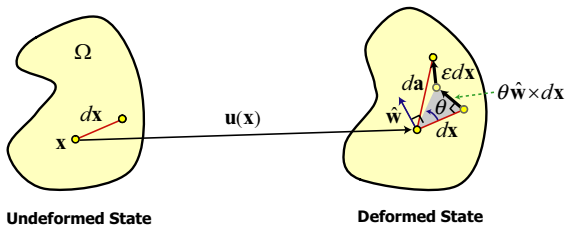

Fig. 1. Kinematics of infinitesimal deformation.

a differential relation that gives the position to which a material point neighboring $\mathbf{x}$ will be mapped by the deformation:

$$d\mathbf{a} = (\mathbf{I} + \nabla\mathbf{u})d\mathbf{x}, \tag{1}$$

where $\nabla\mathbf{u}$ is the Jacobian of $\mathbf{u}$. We are interested in decomposing $\nabla\mathbf{u}$.

The *infinitesimal strain tensor* ε, which measures the change in the squared length of $d\mathbf{x}$ during an infinitesimal deformation (i.e., $\|\nabla\mathbf{u}\| \ll 1$), is defined by

$$\varepsilon \triangleq \tfrac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}).$$

Noting that $\frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}})$ is a meaningful quantity, we can decompose $\nabla\mathbf{u}$ as

$$\nabla\mathbf{u} = \tfrac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}) + \tfrac{1}{2}(\nabla\mathbf{u} - \nabla\mathbf{u}^{\mathrm{T}}) \triangleq \varepsilon + \omega. \tag{2}$$

Interestingly, the skew-symmetric tensor ω is closely related to the curl of the displacement field, $\nabla \times \mathbf{u}$. In fact, ω can be rewritten as

$$\omega = \tfrac{1}{2}(\nabla\mathbf{u} - \nabla\mathbf{u}^{\mathrm{T}}) = \tfrac{1}{2}(\nabla \times \mathbf{u})\times \triangleq \mathbf{w}\times, \tag{3}$$

where $\mathbf{z}\times$ denotes the standard skew symmetric matrix of vector $\mathbf{z}$. Therefore, $\mathbf{w} \triangleq \frac{1}{2}(\nabla \times \mathbf{u})$ can be viewed as a rotation vector that causes rotation of the material points at and near $\mathbf{x}$ by angle $\theta = \|\mathbf{w}\|$ about the unit axis $\hat{\mathbf{w}} = \mathbf{w}/\|\mathbf{w}\|$. ω is called the *infinitesimal rotation tensor*.

By substituting (2) and (3) into (1), we obtain

$$d\mathbf{a} = d\mathbf{x} + \underbrace{\varepsilon d\mathbf{x}}_{\text{strain}} + \underbrace{\theta\hat{\mathbf{w}} \times d\mathbf{x}}_{\text{rotation}},$$

which shows that an infinitesimal deformation consists of a strain and a rotation. This decomposition, illustrated in Fig. 1, has the practical benefit that, for small deformations, it is possible to keep track of the rotation of each material point by calculating the curl of the displacement field, $\mathbf{w} = \frac{1}{2}\nabla \times \mathbf{u}$.

B. Extended Modal Analysis

This section presents how we extend the conventional modal analysis so that it keeps track of the rotation experienced by each material point during deformation. First, we present a brief introductory description of

modal analysis and finite element methods; detailed explanations of these techniques can be found in texts such as [20], [26].

The governing equation for a finite element model is

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}, \tag{4}$$

where $\mathbf{u}(t)$ is a $3n$-dimensional vector that represents the displacements of the n nodes from their original positions, and $\mathbf{F}(t)$ is a vector that represents the external forces acting on the nodes. The mass, damping, and stiffness matrices $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are independent of time and are completely characterized at the rest state, under the commonly adopted assumption (*Rayleigh damping*) that $\mathbf{C} = \xi\mathbf{M} + \zeta\mathbf{K}$, where ξ and ζ are scalar weighting factors.

Modal Displacement. In general, $\mathbf{M}$ and $\mathbf{K}$ are not diagonal, and thus (4) is a coupled system of ordinary differential equations (ODEs). Let Φ and a diagonal matrix Λ be the solution matrices of the generalized eigenvalue problem, $\mathbf{K}\Phi = \mathbf{M}\Phi\Lambda$, such that $\Phi^{\mathrm{T}}\mathbf{M}\Phi = \mathbf{I}$ and $\Phi^{\mathrm{T}}\mathbf{K}\Phi = \Lambda$. Since the columns of Φ form a basis of the $3n$-dimensional space, $\mathbf{u}$ can be expressed as a linear combination of the columns:

$$\mathbf{u}(t) = \Phi\mathbf{q}(t). \tag{5}$$

Here, Φ is the *modal displacement matrix*, of which the i-th column represents the i-th mode shape, and $\mathbf{q}(t)$ is a vector containing the corresponding modal amplitudes as its components. By examining the eigenvalues we can take only dominant m columns of Φ, significantly reducing the amount of computation. In the following, Φ denotes the $3n \times m$ submatrix formed by the above procedure.

Substitution of (5) into (4) followed by a premultiplication of Φ^{T} decouples (4) as

$$\mathbf{M}_{\mathbf{q}}\ddot{\mathbf{q}} + \mathbf{C}_{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{K}_{\mathbf{q}}\mathbf{q} = \Phi^{\mathrm{T}}\mathbf{F}, \tag{6}$$

where $\mathbf{M}_{\mathbf{q}} = \mathbf{I}$, $\mathbf{C}_{\mathbf{q}} = (\xi\mathbf{I} + \zeta\Lambda)$, and $\mathbf{K}_{\mathbf{q}} = \Lambda$ are now all diagonal. $\Phi^{\mathrm{T}}\mathbf{F}$ is called the modal force. The above decoupling allows the motion components due to individual modes to be computed independently and combined by linear superposition.

Modal Rotation. We now develop a procedure to represent the rotational part, $\mathbf{w}(t)$, in terms of $\mathbf{q}(t)$. $\mathbf{w}(t)$ is a $3n$-dimensional vector formed by concatenating all of the 3-dimensional rotation vectors, each of which is formed by taking the curl of the displacement field $\mathbf{u}$ at each node, as described in Section III-A.

For simplicity, we use linear tetrahedral elements in (4). Let $\mathbf{u}_{e,j}$ ($j \in [1,4]$) be the vertex displacement of a tetrahedron Ω_e, and let $\mathbf{u}_e = [\mathbf{u}_{e,1}^{\mathrm{T}}|\mathbf{u}_{e,2}^{\mathrm{T}}|\mathbf{u}_{e,3}^{\mathrm{T}}|\mathbf{u}_{e,4}^{\mathrm{T}}]^{\mathrm{T}}$. Then, the displacement of material point $\mathbf{x} \in \Omega_e$ is given by $\mathbf{u}(\mathbf{x}) = \mathbf{H}_e(\mathbf{x})\mathbf{u}_e$, where $\mathbf{H}_e(\mathbf{x})$ is the linear shape function of the element. Substituting this into (3) yields the rotation vector for $\mathbf{x}$:

$$\mathbf{w}_e(\mathbf{x}) = \tfrac{1}{2}(\nabla\times)\mathbf{H}_e(\mathbf{x})\mathbf{u}_e \triangleq \mathbf{W}_e\mathbf{u}_e. \tag{7}$$

Note that, because $\mathbf{H}_e(\mathbf{x})$ is a linear function of $\mathbf{x}$, $\mathbf{W}_e$ is constant, and thus $\mathbf{w}_e(\mathbf{x})$ is uniform over Ω_e. For the rotation vector of a node, we use the average of the rotation vectors of all the tetrahedra sharing the node.

Based on the above discussion, we can now assemble $\mathbf{W}_e$ of all the elements to form the global matrix $\mathbf{W}$ such that $\mathbf{W}\mathbf{u}(t)$ gives the composite vector $\mathbf{w}(t)$ that we are looking for.[1] Finally, expanding $\mathbf{u}(t)$ with (5) gives

$$\mathbf{w}(t) = \mathbf{W}\Phi\mathbf{q}(t) \triangleq \Psi\mathbf{q}(t). \tag{8}$$

Both $\mathbf{W}$ and Φ are characterized by the deformable mesh at the rest state, and are thus constant over time. Therefore we can precompute Ψ. Equation (8) shows that, as in the displacement (Equation (5)), we can represent the rotational component of deformation in terms of $\mathbf{q}(t)$. We call Ψ the *modal rotation matrix*. It should be noted that both of the modal matrices are meaningful only for moderately small deformations.

IV. Integration of Rotational Parts

Equation (8) provides an efficient way to keep track of the rotations occurring at each node over time. However, such rotations are not yet reflected in the calculation of the displacement field $\mathbf{u}(t)$. Therefore, simulations based on (5), (6), and (8) in Section III-B will not produce proper rotational deformations. In this section, we develop a method to integrate the effect of the rotational part into the calculation of $\mathbf{u}(t)$.

To accommodate large deformations, the stiffness matrix $\mathbf{K}$ in (4) should be replaced by $\mathbf{K}(\mathbf{u})$. Therefore, we must deal with a governing equation of the form,

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u})\mathbf{u} = \mathbf{F}. \tag{9}$$

Let $\mathbf{u}(t) = [\mathbf{u}_i(t)] = [\mathbf{u}_1^{\mathrm{T}}(t)\cdots\mathbf{u}_n^{\mathrm{T}}(t)]^{\mathrm{T}}$. Then the i-th 3-dimensional vector $\mathbf{u}_i(t)$ represents the displacement of the i-th node from its original position, measured in the global coordinate frame. In order to measure the local rotations with respect to the global coordinate frame, we embed a local coordinate frame $\{i\}$ at each node i as shown in Fig. 2, such that at the initial state it is aligned

[1] When assembling $\mathbf{W}$, 3×3 submatrices of $\mathbf{W}_e$ for the rotation vectors of all the tetrahedra sharing a node are not summed up but averaged.

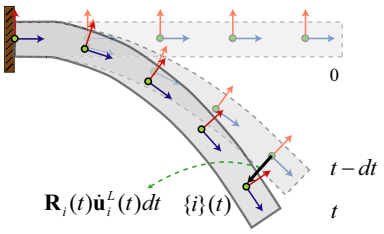

Fig. 2. Local coordinate frames attached to the nodes.

with the global coordinate frame. We use the notation $\{i\}(t)$ to refer to the local coordinate frame at time t.

Let $\mathbf{R}_i(t)$ be the rotation matrix representing the orientation of $\{i\}(t)$, and $\dot{\mathbf{u}}_i^L(t)dt$ be the differential displacement of the i-th node at time $t-dt$ measured from $\{i\}(t-dt)$. Then, the finite displacement $\mathbf{u}_i(t)$ measured from the global coordinate frame is given by

$$\mathbf{u}_i(t) = \int_0^t \mathbf{R}_i(\tau)\dot{\mathbf{u}}_i^L(\tau)d\tau. \tag{10}$$

The above procedure must be carried out for every node. Therefore, we form the block-diagonal matrix $\mathbf{R} = [\delta_{ij}\mathbf{R}_i]$, where $1 \le i, j \le n$ and δ_{ij} is the Kronecker delta. Then, n equations with the form of (10) can be assembled into a single equation,

$$\mathbf{u}(t) = \int_0^t \mathbf{R}(\tau)\dot{\mathbf{u}}^L(\tau)d\tau. \tag{11}$$

This equation shows how the effect of the rotations occurring at the nodal points can be accumulated. The remainder of this section describes the procedure used to compute the above integration.

A. Modal Analysis in Local Coordinate Frames

Equation (11) tells us that, instead of solving (9) for $\mathbf{u}$, we need to convert the equation into a form that can be solved for $\mathbf{u}^L$. By premultiplying both sides of (9) with $\mathbf{R}^{\mathrm{T}}$, we obtain

$$\mathbf{R}^{\mathrm{T}}\left[\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u})\mathbf{u}\right] = \mathbf{R}^{\mathrm{T}}\mathbf{F}. \tag{12}$$

The following two assumptions must then be made to convert (12) to the form shown in (16).

Assumption I: Commutativity in Fine Meshes. We assume that the mesh being simulated is sufficiently fine that the approximation,

$$\mathbf{R}^{\mathrm{T}}\mathbf{M} \approx \mathbf{M}\mathbf{R}^{\mathrm{T}}, \tag{13}$$

is valid. ■

The error associated with the above approximation is related to the orientational differences between neighboring local coordinate frames. To prove that this error decreases as the orientational differences decrease, we need to examine the structure of $\mathbf{M}$. For the simple case where $\mathbf{M}$ is a lumped diagonal mass matrix, the approximation error is zero regardless of $\mathbf{R}$. The proof for the general case is given in Appendix I. Experimental results showed that the approximation error did not significantly impact the visual realism of the simulation, even for coarse meshes.

Differentiating both sides of (11) with respect to time, we obtain $\dot{\mathbf{u}} = \mathbf{R}\dot{\mathbf{u}}^L + \dot{\mathbf{R}}\dot{\mathbf{u}}^L$. Therefore,

$$\mathbf{R}^{\mathrm{T}}\mathbf{M}\ddot{\mathbf{u}} \approx \mathbf{M}\mathbf{R}^{\mathrm{T}}\ddot{\mathbf{u}} = \mathbf{M}\ddot{\mathbf{u}}^L + \mathbf{M}\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}\dot{\mathbf{u}}^L, \tag{14}$$

where $\mathbf{M}\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}\dot{\mathbf{u}}^L$ is the Coriolis force resulting from the rotational movements of the local coordinate frames. If the rotational movements occur at a moderate rate, the Coriolis force is negligible compared to gravity. Thus, we omit the Coriolis force in the subsequent formulation.[2]

Assumption II: Warped Stiffness. We assume that the nonlinear elastic forces can be approximated by

$$\mathbf{K}(\mathbf{u})\mathbf{u} \approx \mathbf{R}\mathbf{K}\mathbf{u}^L \quad \Leftrightarrow \quad \mathbf{R}^{\mathrm{T}}\mathbf{K}(\mathbf{u})\mathbf{u} \approx \mathbf{K}\mathbf{u}^L, \tag{15}$$

which measures linear elastic forces in the local coordinate frames, but resolves them in the global coordinate frame. ■

The above assumption is similar to the *stiffness warping* proposed in [14], where $\mathbf{K}(\mathbf{u})\mathbf{u} \approx \mathbf{R}\mathbf{K}(\mathbf{R}^{\mathrm{T}}\mathbf{a} - \mathbf{x})$. Unlike element-based rotation of elastic forces [4], [15], node-based rotation can yield a non-zero total momentum of elastic forces, and thus a spurious ghost force on a free-floating deformable object. However, the effects of such a ghost force are suppressed by the constraint force acting on a rigid support.

Now, we are ready to approximate (12) by a linear equation for modal analysis in the local coordinate frames. Substituting (14) and (15) into (12), we obtain

$$\mathbf{M}\ddot{\mathbf{u}}^L + \mathbf{C}\dot{\mathbf{u}}^L + \mathbf{K}\mathbf{u}^L = \mathbf{R}^{\mathrm{T}}\mathbf{F}, \tag{16}$$

where we use the proportional damping $\mathbf{C} = \xi\mathbf{M} + \zeta\mathbf{K}$. This linear elastodynamic equation for $\mathbf{u}^L$ is same as (4), except that the external force acting on each node needs to be pre-rotated in accordance with its local coordinate frame. Therefore, it is straightforward to reduce (16) into a set of decoupled ODEs. The modal displacement matrix Φ obtained in Section III-B gives the relationship

$$\mathbf{u}^L(t) = \Phi\mathbf{q}(t), \tag{17}$$

[2]When the coordinate frame movements occur at an extreme rate, the Coriolis force can be taken into account as an external force by replacing RHS of (18) with $\Phi^{\mathrm{T}}(\mathbf{R}^{\mathrm{T}}\mathbf{F}) - \Phi^{\mathrm{T}}(\mathbf{M}\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}\Phi\dot{\mathbf{q}})$. However, the above replacement did not produce noticeable differences for the examples shown in Section VI.

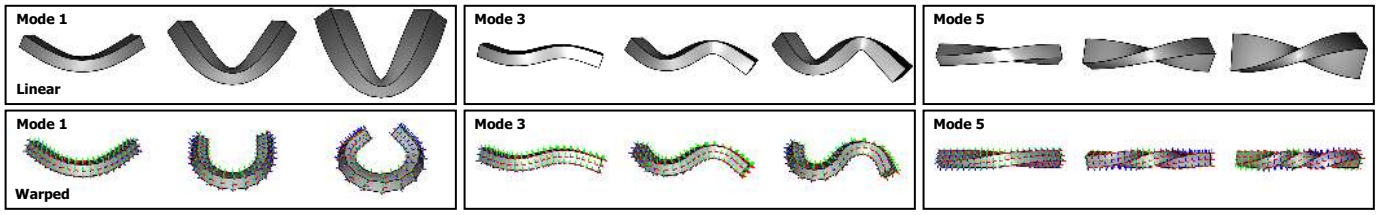

Fig. 3. Evolution of mode shapes in linear modal analysis (top row) and modal warping (bottom row); each box shows snapshots taken at three different amplitudes.

where we use the notation $\mathbf{q}(t)$ instead of $\mathbf{q}^L(t)$ for the sake of readability. Based on this relationship, we can replace $\mathbf{u}^L$ in (16) with $\Phi\mathbf{q}(t)$, and, after premultiplying both sides of the same equation with Φ^{T}, we obtain

$$\mathbf{M_q}\ddot{\mathbf{q}} + \mathbf{C_q}\dot{\mathbf{q}} + \mathbf{K_q}\mathbf{q} = \Phi^{\mathrm{T}}\left(\mathbf{R}^{\mathrm{T}}\mathbf{F}\right). \tag{18}$$

The above decoupled ODEs can be solved numerically using semi-implicit integration.[3] By manipulating (18), we obtain the following expressions for $\mathbf{q}^k = \mathbf{q}(t^k)$ and $\dot{\mathbf{q}}^k = \dot{\mathbf{q}}(t^k)$:

$$\begin{aligned}\mathbf{q}^k &= \alpha\mathbf{q}^{k-1} + \beta\dot{\mathbf{q}}^{k-1} + \gamma\left(\mathbf{R}^{k-1}\Phi\right)^{\mathrm{T}}\mathbf{F}^{k-1}, \\ \dot{\mathbf{q}}^k &= h^{-1}\left[\left(\alpha - \mathbf{I}\right)\mathbf{q}^{k-1} + \beta\dot{\mathbf{q}}^{k-1} + \gamma\left(\mathbf{R}^{k-1}\Phi\right)^{\mathrm{T}}\mathbf{F}^{k-1}\right],\end{aligned} \tag{19}$$

where α, β, and γ are diagonal matrices, the i-th components of which are respectively

$$\alpha_i = 1 - \frac{h^2 k_i}{d_i}, \quad \beta_i = h\left(1 - \frac{hc_i + h^2 k_i}{d_i}\right), \quad \gamma_i = \frac{h^2}{d_i},$$

in which h is the time step size, $d_i = m_i + hc_i + h^2 k_i$ with m_i, c_i, and k_i representing the diagonal entries of $\mathbf{M_q}$, $\mathbf{C_q}$, and $\mathbf{K_q}$, respectively.

B. Formulation of Modal Warping

We now need to evaluate (11) for the finite displacement $\mathbf{u}^k$ at the time step k. When a straightforward numerical integration is employed, accumulation of the numerical errors can give rise to an hysteresis effect such that the deformable solid does not return to the initial state even after all the external forces disappear.

To circumvent such an hysteresis effect, we analytically evaluate (11) by taking a quasi-static approach that ramps $\mathbf{q}(t)$ from 0 to $\mathbf{q}^k$ at each time step k. That is, we use

$$\mathbf{q}(t) = \frac{t}{t^k}\mathbf{q}^k, \quad 0 \le t \le t^k. \tag{20}$$

[3]We employed semi-implicit integration for m decoupled equations because it was easy to implement and the derivation for manipulation constraints became simple. We have not encountered numerical instabilities with a time step size of h = 1/30 second in all our experiments. The possible over-damping effects can be attenuated using theta-integration or Newmark integration. Alternatively, one can use either IIR digital filters suggested in [10] or the closed form solution given in [8]. Finally, we note that all these approaches have the same time complexity because m equations are already decoupled.

Then, the history of $\mathbf{w}(t)$, which determines that of $\mathbf{R}(t)$, is also represented as a linear function.

$$\mathbf{w}(t) = \frac{t}{t^k}\Psi\mathbf{q}^k, \quad 0 \le t \le t^k. \tag{21}$$

Now, $\mathbf{R}(t)$ can be obtained by simply converting $\mathbf{w}(t)$ into the $3n \times 3n$ block-diagonal rotation matrix. Finally, we exploit $\dot{\mathbf{u}}^L(t) = \Phi\dot{\mathbf{q}}(t)$ from (17) and $\dot{\mathbf{q}}(t) = \frac{1}{t^k}\mathbf{q}^k$ from (20) to analytically evaluate (11) as follows:

$$\mathbf{u}^k = \int_0^{t^k} \mathbf{R}(t)\Phi\dot{\mathbf{q}}(t)dt = \tilde{\mathbf{R}}^k\Phi\mathbf{q}^k, \tag{22}$$

where $\tilde{\mathbf{R}}^k \triangleq \frac{1}{t^k}\int_0^{t^k}\mathbf{R}(t)dt$. The procedure for computing $\tilde{\mathbf{R}}^k$ is given in Appendix II.

The above equation implies a new deformation scheme; $\tilde{\Phi}^k \triangleq \tilde{\mathbf{R}}^k\Phi$ can be regarded as a warped version of the original modal basis Φ. The columns of $\tilde{\Phi}^k$ give the mode shapes at the time step k, in which rotations occurred at the nodal points have been accumulated. Fig. 3 shows the evolution of three selected mode shapes over time for the case of a bar. The new method works basically in the same way as linear modal analysis, except that it uses a warped modal basis instead of a fixed linear modal basis.

V. Manipulation Constraints

Thus far, we have discussed the dynamics of an unconstrained elastic body. Motivated by the work of Hauser et al. [8] on positional constraints in a linear modal analysis setting, we extend our deformation scheme to cope with manipulation constraints that allow, for example, dragging/twisting of some nodes to certain positions and/or orientations (see Fig. 7). We formulate these manipulation constraints as hard constraints. Constraints for velocity and acceleration can be developed in a similar way. Note that orientation constraints for a deformable body have not been addressed in previous studies. Such constraints are possible in our formulation because it explicitly takes into account the mean orientation of each node, based on the infinitesimal strain tensor analysis.

A. Position Constraints

Let λ be the number of constrained points, and let $\mathbf{u}_c^k$ be the 3λ-dimensional vector consisting of the desired displacements of the constrained nodes at a time step k. Then, the constraint equation can be written as

$$\mathbf{u}_c^k = \tilde{\Phi}_c^k \mathbf{q}_c^k = \tilde{\mathbf{R}}_c^k \Phi_c \mathbf{q}_c^k, \tag{23}$$

where $\mathbf{q}_c^k$ is the unknown modal amplitude vector, Φ_c is the $3\lambda \times m$ matrix obtained from Φ by taking only the rows for the constrained nodes, and $\tilde{\mathbf{R}}_c^k$ is the $3\lambda \times 3\lambda$ block-diagonal matrix obtained from $\tilde{\mathbf{R}}^k$ by taking only the part corresponding to the constrained nodes.[4] Let the $3n$-dimensional vector $\bar{\mathbf{F}}^{k-1}$ represent the unknown constraint force measured in the global coordinate frame. Then, $\mathbf{q}_c^k$ should satisfy not only (23) but also (19) when this additional force is applied. That is,

$$\begin{aligned}\mathbf{q}_c^k &= \alpha \mathbf{q}^{k-1} + \beta \dot{\mathbf{q}}^{k-1} + \gamma \big(\mathbf{R}^{k-1}\Phi\big)^{\mathrm{T}} \big(\mathbf{F}^{k-1} + \bar{\mathbf{F}}^{k-1}\big) \\ &\triangleq \mathbf{q}_u^k + \gamma \big(\mathbf{R}^{k-1}\Phi\big)^{\mathrm{T}} \bar{\mathbf{F}}^{k-1},\end{aligned} \tag{24}$$

where $\mathbf{q}_u^k \triangleq \alpha \mathbf{q}^{k-1} + \beta \dot{\mathbf{q}}^{k-1} + \gamma (\mathbf{R}^{k-1}\Phi)^{\mathrm{T}} \mathbf{F}^{k-1}$ is the modal amplitude vector for the unconstrained case, i.e., $\mathbf{q}^k$ in (19).

The forces do not need to be exerted only at the constrained nodes, because exerting forces at some unconstrained nodes can still cause the constrained nodes to be positioned at the specified locations. We will refer to the nodes at which forces are exerted as *exercised* nodes. When we directly drag a set of nodes, for example, the exercised nodes are identical to the constrained nodes. In general, however, they can be different.

Let μ be the number of exercised nodes. In $\bar{\mathbf{F}}^{k-1}$, the portion corresponding to the unexercised nodes should be zero. Let $\mathbf{F}_x^{k-1}$ be the 3μ-dimensional vector consisting only of the constraint forces acting on the exercised nodes, which can be obtained by removing the 3-dimensional vectors corresponding to the unexercised nodes from $\bar{\mathbf{F}}^{k-1}$. Then, we can rewrite (24) in terms of $\mathbf{F}_x^{k-1}$,

$$\mathbf{q}_c^k = \mathbf{q}_u^k + \gamma \big(\mathbf{R}_x^{k-1}\Phi_x\big)^{\mathrm{T}} \mathbf{F}_x^{k-1}, \tag{25}$$

where Φ_x is the $3\mu \times m$ matrix obtained from Φ by taking only the rows for the exercised nodes, and the $3\mu \times 3\mu$ block-diagonal matrix $\mathbf{R}_x^{k-1}$ is obtained from $\mathbf{R}^{k-1}$ by taking only the part corresponding to the exercised nodes. Finally, substituting (25) into (23) and manipulating the resulting expression, we obtain the equation for the constraint force:

$$\mathbf{F}_x^{k-1} = \mathbf{R}_x^{k-1} \mathbf{A}_p^{\dagger} \mathbf{b}_p, \tag{26}$$

where $\mathbf{A}_p = \tilde{\mathbf{R}}_c^k \Phi_c \gamma \Phi_x^{\mathrm{T}}$, $\mathbf{b}_p = \mathbf{u}_c^k - \tilde{\mathbf{R}}_c^k \Phi_c \mathbf{q}_u^k$, and $(\cdot)^{\dagger}$ denotes the pseudo-inverse of a matrix. This constraint force can now be applied to the exercised nodes through (25) to yield the desired modal amplitude vector.

We now examine the computational complexity of (26). Since $\mathbf{A}_p$ is time-dependent, the pseudo-inverse of $\mathbf{A}_p$ must be computed at every time step. Fortunately, we can decompose $\mathbf{A}_p$ into time-dependent and time-independent parts, namely $\mathbf{A}_p = (\tilde{\mathbf{R}}_c^k)(\Phi_c \gamma \Phi_x^{\mathrm{T}})$, making it possible to compute its pseudo-inverse using $\mathbf{A}_p^{\dagger} = (\Phi_c \gamma \Phi_x^{\mathrm{T}})^{\dagger} (\tilde{\mathbf{R}}_c^k)^{-1}$. The first part of $\mathbf{A}_p^{\dagger}$ is time-independent, and hence can be precomputed at the constraint initiation stage. The second part is time dependent and therefore must be computed at runtime; however, this entails only a small computational load because $\tilde{\mathbf{R}}_c^k$ is (3×3)-block-diagonal.

B. Orientation Constraints

Orientation constraints can be implemented in a similar way to the position constrains. Let η be the number of constrained nodes, and let the 3η-dimensional vector $\mathbf{w}_c^k$ represent the desired rotations of the constrained nodes at a time step k. Then, the constraint equation can be written as

$$\mathbf{w}_c^k = \Psi_c \mathbf{q}_c^k, \tag{27}$$

where $\mathbf{q}_c^k$ is the unknown modal amplitude vector and Ψ_c is the $3\eta \times m$ matrix obtained from the modal rotation matrix Ψ by taking only the rows corresponding to the constrained nodes. Then, as in the position constraint case, $\mathbf{q}_c^k$ should simultaneously satisfy (25) and (27). By manipulating these two equations, we obtain the equation for the constraint force:

$$\mathbf{F}_x^{k-1} = \mathbf{R}_x^{k-1} \mathbf{A}_o^{\dagger} \mathbf{b}_o, \tag{28}$$

where $\mathbf{A}_o = \Psi_c \gamma \Phi_x^{\mathrm{T}}$ and $\mathbf{b}_o = \mathbf{w}_c^k - \Psi_c \mathbf{q}_u^k$. Unlike the position constraint case, $\mathbf{A}_o$ is time-invariant so its pseudo-inverse can be precomputed at the constraint initiation stage. Finally, we can apply the above constraint force to the exercised nodes through (25) to obtain the desired modal amplitude vector.

[4] We note that the content of $\tilde{\mathbf{R}}^k$ ($\tilde{\mathbf{R}}_c^k$ is its submatrix) used in this section may differ from that of $\tilde{\mathbf{R}}^k$ appearing in (22), since extra movements may need to be incurred to realize the constraints. We propose three ways of treating the problem: (a) employ the Newton-Rhapson method; (b) approximate $\tilde{\mathbf{R}}_c^k$ from $\tilde{\mathbf{R}}^k$ of (22); or (c) use a slightly less accurate version of (22), i.e., $\mathbf{u}^k = \tilde{\mathbf{R}}^{k-1}\Phi \mathbf{q}^k$. Each of these methods has its drawbacks; method (a) can require longer computation times, method (b) can potentially cause oscillations, and method (c) can make the simulation off-phase by one time step. We found method (b) to be a reasonable choice because, during experiments, no noticeable oscillations has been observed.

C. Mixed Constraints

When one set of nodes is position-constrained and another (not necessarily disjoint) set is orientation-constrained, the constraint force should simultaneously satisfy both types of constraint. A simple approach would be to use an augmented formulation that combines (26) and (28):

$$\mathbf{F}_x^{k-1} = \mathbf{R}_x^{k-1} \begin{bmatrix} \mathbf{A}_p \\ \mathbf{A}_o \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{b}_p \\ \mathbf{b}_o \end{bmatrix}.$$

However, this approach does not allow precomputation of the pseudo-inverse because $[\mathbf{A}_p^{\mathrm{T}} \ \mathbf{A}_o^{\mathrm{T}}]^{\mathrm{T}}$ is time-dependent.

To isolate the precomputable part, we employ a task-priority approach [16] in which the position constraints are regarded as the primary task and the orientation constraints as the secondary task (or vice versa, depending on the situation). Letting $\mathbf{f}_p = \mathbf{A}_p^\dagger \mathbf{b}_p$, the constraint force can be written as

$$\mathbf{F}_x^{k-1} = \mathbf{R}_x^{k-1} \left\{ \mathbf{f}_p + \left[\mathbf{A}_o(\mathbf{I} - \mathbf{A}_p^\dagger \mathbf{A}_p)\right]^\dagger \left[\mathbf{b}_o - \mathbf{A}_o \mathbf{f}_p\right] \right\}, \quad (29)$$

which causes the solution satisfying the position constraints is found first, and then the solution optimally satisfying the orientation constraints is searched for within the null space of the position constraints. Note that $\mathbf{A}_p^\dagger \mathbf{A}_p$ is time-independent because the time-dependent parts cancel each other, and hence $\left[\mathbf{A}_o(\mathbf{I} - \mathbf{A}_p^\dagger \mathbf{A}_p)\right]^\dagger$ can be precomputed. Consequently, the only nontrivial computation remaining in the calculation of (29) is to compute the inverse of $\tilde{\mathbf{R}}_c^k$, which appears in $\mathbf{f}_p = \mathbf{A}_p^\dagger \mathbf{b}_p = (\Phi_c \gamma \Phi_c^{\mathrm{T}})^\dagger (\tilde{\mathbf{R}}_c^k)^{-1} \mathbf{b}_p$.

D. Static Position Constraints

In the above description of manipulation constraints, every positional or rotational displacement is measured relative to the frame of reference [25], which was introduced in Section II. The *static position constraints*, that makes a set of nodes be fixed at the initial locations with respect to the frame of reference, is not implemented in terms of the manipulation constraints. Constraints of this type are realized by simply omitting the corresponding DOFs in the governing equation and setting the displacements to zero.

VI. Experimental Results

Our deformation scheme is implemented as an Alias® MAYA™ plugin for a Microsoft® Windows^{XP} environment, and also as a stand-alone application to exploit programmable graphics hardware through *n*VIDIA® Cg and Microsoft® DirectX® API. Tetrahedral meshes were generated using the public domain software NETGEN. To obtain the m dominant eigenvalues of large sparse square matrices and the corresponding eigenvectors, we used the MATLAB® built-in C++ math function `eigs`, which is based on the ARPACK [13] eigenvalue solver. All experiments were performed on a PC with an Intel® Pentium®4 3.2GHz processor, 1GB memory, and an *n*VIDIA® GeForce® FX 5900 Ultra 256MB graphics card. We used the time step size of $h = 1/30$ second in all experiments reported in this section. Model statistics and performance data are summarized in Table I. Animation clips are available at `http://graphics.snu.ac.kr/~mgchoi/modal_warping`.

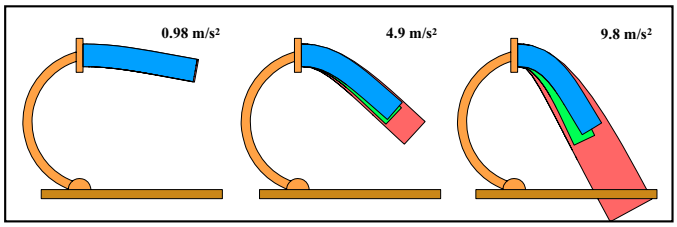

Fig. 4. A bar deformed by modal analysis (red), by modal warping (blue), and by nonlinear FEM (green) under gravity of different magnitudes.

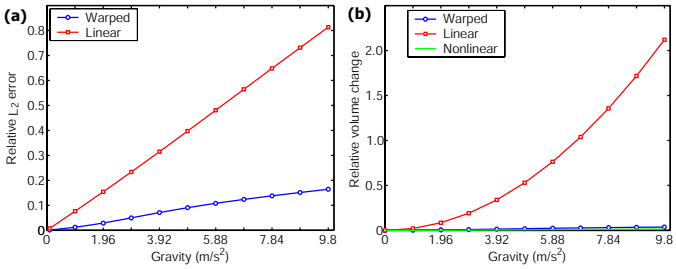

Fig. 5. Error analysis of the bar shown in Fig. 4. (a) the relative L_2 displacement field error and (b) the relative volume change with respect to the initial volume.

Comparison to Other Methods. This experiment is to compare the results generated by linear modal analysis, modal warping, and nonlinear FEM. We ran the three methods to deform a long bar under different gravities. As for the nonlinear FEM [17], we employed explicit integration and used the time step size $h = 0.001$ seconds for numerical stabilities. Fig. 4 shows the snapshots taken at the equilibrium states of the bar.

Fig. 5 (a) shows the plot of the relative L_2 displacement field error versus gravitational magnitude. We took the result produced by nonlinear FEM as the ground truth. The relative error in modal warping is smaller than that in linear modal analysis although it increases as the gravitational magnitude increases. Fig. 5 (b) is the plot of the relative volume change with respect to the

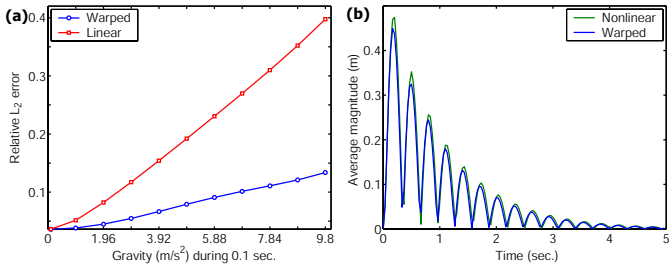

Fig. 6. Error analysis of free vibration: (a) the relative L_2 displacement field error summed over space and time, and (b) the average magnitude of nodal displacements over time.

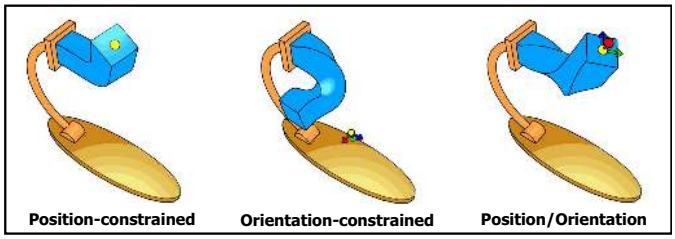

Fig. 7. A bar manipulated with a position constraint (left), an orientation constraint (middle), and a position/orientation constraint (right). The position constraints are represented by yellow spheres and the orientation constraints are represented by RGB axes.

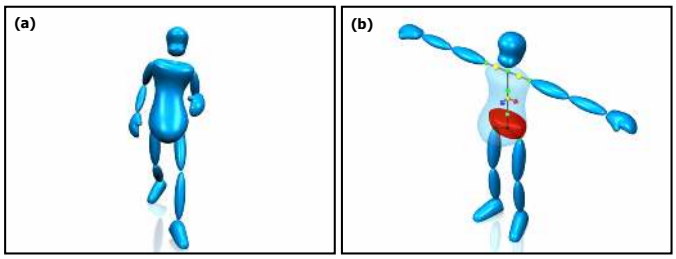

Fig. 8. Constraint-driven animation of a character with one deformable part (the torso).

initial volume. It shows that the relative volume change in modal warping is almost identical to that in nonlinear FEM. Even though Fig. 5 (a) shows modal warping produces non-negligible L_2 displacement field errors, it was not easy to visually discriminate between the results produced by modal warping and nonlinear FEM, unless the results were seen overlayed. However, the effects due to the volume changes were clearly noticeable.

We also conducted a dynamic analysis while the bar makes free vibration. We applied the gravity (of different magnitudes as in the above experiments) during only the first 0.1 seconds of the simulation. Fig. 6 (a) shows the plot of the relative L_2 displacement field error summed over space and time. Fig. 6 (b) is the time-series plot of the average magnitude of nodal displacements in the case of gravitational magnitude $9.8m/s^2$, in which we can observe a subtle difference in the frequency of oscillation. It is interesting to note that, if measured relative to the error of linear modal analysis, the error of modal warping in the dynamic analysis (Fig. 6 (a)) is larger than that in the static analysis (Fig. 5 (a)). It results from the aforementioned difference in the frequency of oscillation.

Manipulation Test. This experiment demonstrates the manipulation capability of our technique. Fig. 7 shows, from left to right, the resultant deformations in the cases of only position constraints, only orientation constraints, and both position and orientation constraints. For the case of position constraints, the constrained node was identical to the exercised node. For the case of orientation constraints, however, the set of exercised nodes had to be extended to include nodes neighboring the constrained node.

Manipulation Constraints for Animating Deformable Body Parts. To demonstrate how the manipulation constraints can be used to animate deformable parts of a character, we simulated a character whose only deformable part was its potbellied torso (Fig. 8(a)). As the character made a dance motion, the potbelly made a dynamic passive deformation, excited by the gross motion of the character as in [10]. All the mesh nodes contained in the rigid pelvis at the initial setup were static position constrained, and thus their movement coincided with that of the pelvis. As shown in Fig. 8(b), the deformable solid is attached to the skeleton by two position constraints (the yellow spheres) and one position/orientation constraint (the RGB axes).

Manipulation Constraints for Motion Retargeting. The manipulation constraints can also be used to retarget a motion of an articulated character to that of a deformable character. To demonstrate this, we consider two examples. In the first example, a jumping motion of an articulated character is retargeted to a jelly box, as shown in Fig. 9(a). As in the character considered above, the nodes contained in the pelvis are static position constrained. The motion of the jelly box is driven by the movements of the feet and head of the articulated character; to implement this, a node corresponding to the middle of the two feet is selected and position/orientation-constrained to follow the average movement of the feet, and a node corresponding to the forehead is also position/orientation-constrained to follow the movement of the head. Three snapshots taken during this experiment are shown in Fig. 9(a). For comparison, we also applied the traditional modal analysis to this case (see Fig. 9(b)).

TABLE I

MODEL STATISTICS AND PERFORMANCE DATA.

Example	Fig.	Model statistics					Constr.		Precomputation (sec)		Computation (sec/fr)		FPS	
		Vertices	Faces	Nodes	Elements	Modes	λ	η	FEM	MA	ODE	Constr.	Maya	Cg
Bar	4 & 7	354	352	99	240	8	1	1	0.046	0.063	0.001	0.001	60.0	·
Potbelly	8	1026	1056	363	1110	16	3	1	0.062	0.484	0.001	0.001	60.0	·
Jelly Box	9(a)	1642	1640	400	1440	32	2	2	0.062	1.156	0.001	0.001	60.0	·
Flubber	9(c)	2802	2800	552	1513	64	6	1	0.078	2.062	0.001	0.001	60.0	·
Dinosaur	10	28098	56192	1883	5484	8	1	1	0.312	1.422	0.002	0.001	11.9	103.8

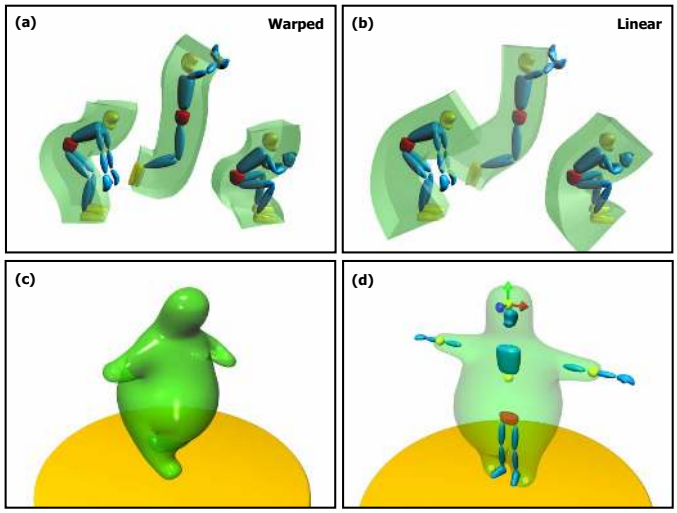

Fig. 9. Constraint-based motion retargeting.

In the second example, we applied a dance motion to the flubber shown in Fig. 9(c). Because this character has a more articulated shape than that in the previous example, more constraints are required to properly map between the articulated and deformable characters. We placed one position/orientation constraint at the head, and five position constraints at the torso, elbows, and feet (see Fig. 9(d)). For the flubber, we used a larger number of deformation modes (64 modes) than in experiments described above; this was necessary to accommodate the wider range of shape variations due to the increased number of constraints.

Simulation of Large Models. When the modal warping technique is applied to a large model such as the dinosaur model shown in Fig. 10, simulating the deformation is not the bottleneck; surprisingly, the dynamic update of the vertex coordinates for display is the slowest procedure. To achieve real-time simulation of the model, we employed programmable graphics hardware as in [10]. The main CPU is devoted to simulating the deformable model. The GPU updates each vertex using both the modal amplitude vector supplied from the CPU and the per-vertex data residing in the video memory of the graphics hardware. In our implementation, the per-vertex data consists of the initial position of the vertex along with an additional $2m$ three-dimensional vectors for the modal displacements and rotations of the vertex. Unlike [10], our method does not require any special considerations on vertex normal corrections because the per-vertex rotation vector is explicitly available to the vertex program (see Appendix III). However, our vertex program requires extra instructions for converting the rotation vector into the rotation matrix. Given the ever-increasing capabilities of graphics hardware technology, we expect that hardware restrictions on the number of instructions will soon be lifted.

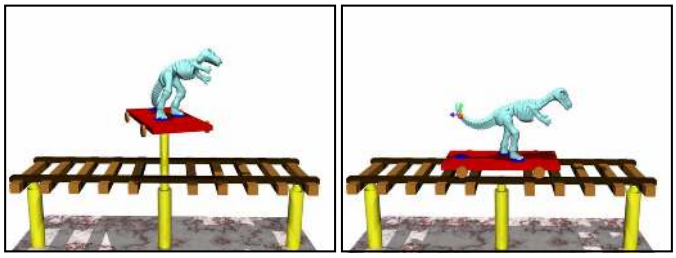

Fig. 10. Dynamic deformation and manipulation of a dinosaur.

To test our approach on a large model, we applied our hardware implementation to the rubber dinosaur model previously used by James and Fatahalian [9]. The mesh for finite element modeling consists of 5,484 tetrahedral elements and 1,883 nodal points, and the mesh for the final display consists of 56,192 faces and 28,098 vertices. The total precomputation time for the finite element method and the modal analysis was less than 2 seconds, and the simulation, including the display, ran at about 100 fps. The result was quite realistic, even for cases involving large deformations. Using our method, the types of interactions allowed during runtime did not need to be restricted; for example, the tail of the dinosaur could be manipulated interactively.

VII. CONCLUSION

The present work extends traditional linear modal analysis to create a novel deformation technique that combines the merits of this type of analysis, in particular its ability to give real-time performance [19], [10], with

the ability to accommodate large rotational deformations. An interesting feature of our technique is that it supports both position and orientation constraints, and hence could be used for interactively manipulating the shape of a deformable solid. The constraints can also be used for some less obvious but very useful purposes, such as to model articulated deformable characters or to drive a keyframe animation such that the animator controls the movement of only a few constrained points then the technique generates the movement of all the nodal points. We expect the deformation technique proposed here will prove useful in many application areas, including computer games and character animation.

Limitations and Future Work. A shortcoming of our method is that, although it adequately accounts for the rotational component, it does not preserve the volume. Therefore, deformations involving a large degree of stretching or compression may generate noticeable artifacts. Another shortcoming of our method is that, even when animating a single undamped mode, the vibration frequency is constant independent of the motion. These behaviors are obvious consequences of using a strain tensor consisting of only linear terms. More accurate modeling of such deformations would require the use of a nonlinear strain tensor.

Further research is needed to address another limitation of our technique. Currently, our technique supports only constrained deformable objects attached to rigid supports. We plan to extend our work for free-floating deformable objects in the future by combining the modal warping framework with rigid body simulation as Hauser et al. [8] did for free-floating objects that undergo moderately small deformations. Collision detection and response among deformable solids and their surrounding environment could also be handled as in [8].

ACKNOWLEDGEMENTS

This research was supported by Korea Ministry of Information and Communication. This research was also partially supported by Automation and Systems Research Institute at Seoul National University, and the Brain Korea 21 Project.

APPENDIX I
ANALYSIS OF EQUATION (13)

The mass matrix, assembled from linear tetrahedral elements, can be written as $\mathbf{M} = [m_{ij}\mathbf{I}]$ for $1 \le i, j \le n$, where $\mathbf{I}$ is the 3×3 identity matrix, and m_{ij} is non-zero if and only if the i-th and j-th nodes are connected in the mesh. Remembering that $\mathbf{R} = [\delta_{ij}\mathbf{R}_i]$, we can expand the approximation error as

$$\begin{aligned}\mathbf{R}^{\mathrm{T}}\mathbf{M} - \mathbf{M}\mathbf{R}^{\mathrm{T}} &= \left[\delta_{ij}\mathbf{R}_i^{\mathrm{T}}\right]\left[m_{ij}\mathbf{I}\right] - \left[m_{ij}\mathbf{I}\right]\left[\delta_{ij}\mathbf{R}_i^{\mathrm{T}}\right] \\ &= \left[m_{ij}\mathbf{R}_i^{\mathrm{T}}\right] - \left[m_{ij}\mathbf{R}_j^{\mathrm{T}}\right] \\ &= \left[m_{ij}(\mathbf{R}_i - \mathbf{R}_j)^{\mathrm{T}}\right],\end{aligned}$$

Here, the error in each block is dependent on the orientational difference, and thus it decreases with increasing mesh resolution. As one refines the mesh, the number of non-zero blocks also increases. However, this increase is cancelled out by the decrease of m_{ij} because the total mass $\sum m_{ij}$ is independent of the mesh resolution. Therefore, the matrix norm of the approximation error decreases with increasing mesh resolution.

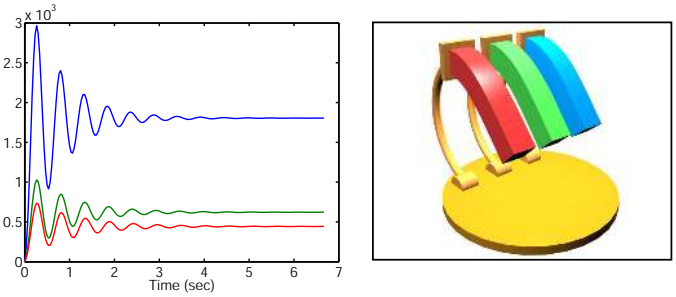

Fig. 11. $\|\mathbf{R}^{\mathrm{T}}\mathbf{M} - \mathbf{M}\mathbf{R}^{\mathrm{T}}\|$ in three different mesh resolutions: $11\times 3\times 3$ (red), $19\times 4\times 4$ (green), and $21\times 5\times 5$ (blue)

To assess the approximation error, we prepared three meshes of different resolutions for the same long bar: $11\times 3\times 3$, $19\times 4\times 4$, and $21\times 5\times 5$. Fig. 11 shows the approximation error $\|\mathbf{R}^{\mathrm{T}}\mathbf{M} - \mathbf{M}\mathbf{R}^{\mathrm{T}}\|$ with respect to the total mass of the deformable body. Even with the coarsest mesh, the error is within 0.37% of the total mass.

APPENDIX II
COMPUTATION OF EQUATION (22)

To compute $\tilde{\mathbf{R}}^k$ in (22), we first convert the rotation vector $\mathbf{w}_i(t)$ of each node into the rotation matrix $\mathscr{R}(\mathbf{w}_i(t))$. For this conversion we employ Rodrigues' formula [21] that expresses the rotation matrix in terms of the angle and the unit axis of rotation. Let $\mathbf{w}_i^k$ be the i-th three-dimensional vector of $\Psi\mathbf{q}^k$. Then, $\mathbf{w}_i(t) = \tau\mathbf{w}_i^k$, where $\tau = t/t^k$. Rodrigues' formula gives

$$\mathscr{R}(\tau\mathbf{w}_i^k) = \mathbf{I} + (\hat{\mathbf{w}}_i^k\times)\sin\|\tau\mathbf{w}_i^k\| + (\hat{\mathbf{w}}_i^k\times)^2(1-\cos\|\tau\mathbf{w}_i^k\|),$$

where $\hat{\mathbf{w}}_i^k = \mathbf{w}_i^k/\|\mathbf{w}_i^k\|$. Now, we integrate both sides of this equation from $\tau = 0$ to 1, Then, $\tilde{\mathbf{R}}_i^k \triangleq \int_0^1 \mathscr{R}(\tau\mathbf{w}_i^k)d\tau$ is given by

$$\tilde{\mathbf{R}}_i^k = \left[\mathbf{I} + (\hat{\mathbf{w}}_i^k\times)\frac{1-\cos\|\mathbf{w}_i^k\|}{\|\mathbf{w}_i^k\|} + (\hat{\mathbf{w}}_i^k\times)^2\left(1 - \frac{\sin\|\mathbf{w}_i^k\|}{\|\mathbf{w}_i^k\|}\right)\right].$$

Finally, the composite block-diagonal rotation matrix for $\mathbf{w}^k = [\mathbf{w}_i^k]$ can be constructed by $\tilde{\mathbf{R}}^k = [\delta_{ij}\tilde{\mathbf{R}}_i^k]$.

APPENDIX III
VERTEX PROGRAM IN CG

```
// float3 phi# is the #-th modal displacement.
// float3 psi# is the #-th modal rotation.
// uniform float4 q contains 4 modal amplitudes.
float3 u = phi1*q.x+phi2*q.y+phi3*q.z+phi4*q.w;
float3 w = psi1*q.x+psi2*q.y+psi3*q.z+psi4*q.w;

// Coefficients for Rodrigues' formula
float  w_len = length(w);
float3 w_hat = normalize(w);
float  s, c; sincos(w_len, s, c);
float  c1    = (1-c)/w_len;
float  c2    = 1 - s/w_len;

// Position correction: \tidle{R}(w) * u
float3 P   = position + u;
P = P + cross(w_hat,u)*c1
      + cross(w_hat,cross(w_hat,u))*c2;

// Normal correction: R(w) * N
float3  N = normal;
N = N + cross(w_hat,N)*s
      + cross(w_hat,cross(w_hat,N))*(1-c);
```

REFERENCES

[1] T. Belytschko, W. Kam, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, Ltd., New York, NY, 2003.

[2] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2002)*, 21(3):586–593, 2002.

[3] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. *Computer Graphics (Proc. ACM SIGGRAPH 2001)*, 35:31–36, 2001.

[4] O. Etzmuss, M. Keckeisen, and W. Straβer. A fast finite element solution for cloth modeling. In *Proc. Pacific Graphics 2003*, pages 244–251, 2003.

[5] C. A. Felippa. A systematic approach to the element-independent corotational dynamics of finite elements. Technical Report CU-CAS-00-03, Center for Aerospace Structures, Colorado, 2000.

[6] S. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Lab., Cambridge, MA, 1997.

[7] E. Grinspun, P. Krysl, and P. Schröder. CHARMS: A simple framework for adaptive simulation. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2002)*, 21(3):281–290, 2002.

[8] K. K. Hauser, C. Shen, and J. F. O'Brien. Interactive deformation using modal analysis with constraints. In *Proc. Graphics Interface 2003*, pages 247–255, 2003.

[9] D. L. James and K. Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2003)*, 22(3):879–887, 2003.

[10] D. L. James and D. K. Pai. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2002)*, 21(3):582–585, 2002.

[11] D. L. James and D. K. Pai. BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2004)*, 23(3), 2004.

[12] P. G. Kry, D. L. James, and D. K. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *Proc. ACM SIGGRAPH Symp. Computer Animation 2002*, pages 153–159, 2002.

[13] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, PA, 1998.

[14] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proc. ACM SIGGRAPH Symp. Computer Animation 2002*, pages 49–54, 2002.

[15] M. Müller and M. Gross. Interactive virtual materials. In *Proc. Graphics Interface 2004*, pages –, 2004.

[16] Y. Nakamura and H. Hanafusa. Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986.

[17] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. *Computer Graphics (Proc. ACM SIGGRAPH '99)*, 33(4):137–146, 1999.

[18] J. F. O'Brien, C. Shen, and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. pages 175–182, 2002.

[19] A. Pentland and J. Williams. Good vibrations: Model dynamics for graphics and animation. *Computer Graphics (Proc. ACM SIGGRAPH '89)*, 23(3):207–214, 1989.

[20] A. A. Shabana. *Theory of Vibration, Volume II: Discrete and Continuous Systems*. Springer-Verlag, New York, NY, 1990.

[21] A. A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, 1998.

[22] J. Stam. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum (Proc. EUROGRAPHICS '97)*, 16(3):159–164, 1997.

[23] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics (Proc. ACM SIGGRAPH '88)*, 22(4):269–278, 1988.

[24] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics (Proc. ACM SIGGRAPH '87)*, 21(4):205–214, 1987.

[25] D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics & Applications*, 8(6):41–51, 1988.

[26] O. C. Zienkiewicz. *The Finite Element Method*. McGraw-Hill Book Company (UK) Limited, Maidenhead, Berkshire, England, 1977.

Min Gyu Choi received the BS, MS, and PhD degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1996, 1998, and 2003, respectively. He is currently a postdoctoral fellow in the School of Electrical Engineering and Computer Science at Seoul National University (SNU). His primary research interests are flexible body animation, motion planning, motion editing, and motion retargeting.

Hyeong-Seok Ko received the BS and MS degrees in computer science from SNU, Korea, in 1985 and 1987, respectively, and PhD degree in computer science from University of Pennsylvania in 1994. He is an associate professor in the School of Electrical Engineering and Computer Science at Seoul National University (SNU), Korea. He has been working at SNU since 1994. His recent research interest includes cloth simulation, fluid simulation, facial animation, hair modeling, flexible body animation, and motion retargeting.