Interactive Shape Interpolation through Controllable Dynamic Deformation

Jin Huang, Yiying Tong, Kun Zhou, Hujun Bao and Mathieu Desbrun

Abstract—In this paper, we introduce an interactive approach to generate physically-based shape interpolation between poses. We extend linear modal analysis to offer an efficient and robust numerical technique to generate physically-plausible dynamics even for very large deformation. Our method also provides a rich set of intuitive editing tools with real-time feedback, including control over vibration frequencies, amplitudes, and damping of the resulting interpolation sequence. We demonstrate the versatility of our approach through a series of complex dynamic shape interpolations.

Index Terms—Deformation Gradient, Shape Interpolation, Space-Time Constraints, Modal Analysis

1 Introduction

In computer animation of characters or deformable objects, interpolating between two given poses (key frames) of a same mesh is a ubiquitous task. One must not only produce visually-pleasing deformations between shapes, but also create complex dynamic idiosyncracies along the way to increase realism and visual impact. Offering an easy-to-use, interactive framework for a dynamic shape interpolation is currently, however, considered difficult.

1.1 Problem Statement

In this paper, we will deal with 3D objects, each represented as a simplicial complex (x, C), where C encodes the connectivity of the tetrahedral mesh and $x = (x_1^t, \cdots, x_n^t)^t \in R^{3n}$ denotes the shape, *i.e.*, the set of positions of the mesh vertices. We assume that an object (r, C), where r is its *rest shape*, is provided by the user, along with two poses x_A, x_B of this 3D object. We wish to evaluate a physically-based shape deformation $x(t)$ with $t \in [0, T]$, such that $x(0) = x_A$ and $x(T) = x_B$ while offering intuitive and interactive control over the dynamics throughout the interpolation sequence.

To address this problem, we will introduce a method that converts the deformation gradient between a shape and its rest position into local rotations and strains, and performs spectral analysis to efficiently interpolate two shapes with a complete control over the motion. Working in the space of rotations and strains will prove to handle extremely large deformation correctly, making our approach very robust, yet interactive.

- *J. Huang, K. Zhou and H. Bao are with State Key Lab of CAD&CG, Zhejiang University*
- *Y. Tong is with the Department of Computer Science and Engineering, Michigan State University.*
- *M. Desbrun is with the Computing and Mathematical Sciences department, California Institute of Technology.*

1.2 Previous Work

Pose interpolation has been investigated in a number of contexts. We briefly review the main approaches next.

Purely Geometric Interpolation Shape interpolation methods provide fast and reasonable inbetweening of two given geometric shapes, albeit without control over dynamical effects. For example, "as-rigid-as-possible" shape interpolation establishes compatible triangulation of 2D or 3D shapes, and creates per-simplex interpolations of the corresponding simplices based on a *geometric decomposition of the transformation matrix* into a rotation and a stretching matrix. The final vertex paths are computed through a global best fit of these per-simplex transformations [1]. In a more recent work, Kircher et al. [2] achieve a wider range of geometric interpolations through blending of differential (rotation- and translation-invariant) representations of poses. Geometric modeling in "shape space" is another purely geometric approach to shape interpolation, where a notion of geodesics in shape space is used to provide an as-isometric-as-possible path between two poses [3].

Physically-based Interpolation If material properties are known, one can theoretically solve for the optimal deformation between two given poses of an elastic object through space-time constraints and/or optimal control methods [4], [5]. However, nonlinearities induced by the laws of physics render these approaches dramatically slower than purely geometric methods for complex objects (if not intractable). More recently, an approach based on coarse and meshless modeling [6] of deformable shapes was introduced to produce smooth interpolations at interactive frame rates through a minimization of rigidity and volume-preservation energies. However, the results are visually similar to purely geometric methods, producing little to no vibrations due to the choice of mostly geometrically-motivated energies. Optimal control was also explored lately to adjust a motion to a desired trajectory in real-time [7], but it did

Fig. 1. Our shape interpolation method can interactively provide a dynamic motion between poses as demonstrated in this 4-keyframe interpolation on a raptor model (rest shape, top left). While smooth interpolation can be obtained automatically (shadow shapes, yellow), the user has control over the dynamics in realtime, allowing easy and interactive editing of frequency, amplitude, and damping of vibration modes to enhance motion complexity.

not address the challenging problem of getting the initial trajectory connecting the input key frames at interactive rates. A *reduced* optimal control was also recently proposed to give plausible dynamical interpolation [8]. However, this approach requires significant computations each time a new keyframe is chosen, preventing realtime user interaction.

Modal Analysis Although not directly applicable to the problem of pose interpolation per se, a particularly efficient and natural way to model physical behavior is through modal analysis [9], [10], which decomposes the space of deformations into a set of vibration modes. Note that linear modal analysis is only physically valid for small perturbation around the pose over which *spectral analysis* of the stiffness and mass matrices is performed. However, a corotational method ("Modal Warping") was proposed to gracefully extend the original approach to large deformations [11], at the price of an approximate time integration to compute positions. Unfortunately, modal analysis has been used primarily for forward physical simulation instead of interpolation between given key frames. Recent model reduction methods [12], [13] are purely data-driven, with no explicit modeling or control of dynamic properties, such as mass, damping, frequency etc. Closer to our concern is a recent approach coined *Wiggly Splines* [14], which provides a link between traditional animation splines and vibration modes. With their framework, assuming that vibration modes of an object are known (via either procedural or example-based methods), an animator can design a motion from one pose to another containing intentional vibrations whose frequency, damping, and relative phase can be tweaked as desired for cinematic effect. While this method offers intuitive control of the deformation and its dynamics, the linear space of deformations that the authors restrict the approach to seems mostly amenable, as is, to wiggly motions.

1.3 Rationale and Contributions

We depart from previous methods by proposing an approach that draws both on physically-based (in particular, through spectral motion decomposition) *and* geometric methods (through a geometric decomposition of eigenmodes). Building upon the equations of linear elasticity that govern small deformations, our hybrid technique treats linear deformation modes as tangent vectors in a *rotation-strain space*, which describes the deformed object through its deformation gradient field represented as the product of a local rotation and a material stretch tensor. Linear combinations of modes in this new representation produce physically-plausible *very large elastic deformations*, extending the domain of validity of Modal Warping even further. As our approach allows a simple map from the shape to the modes, we also contribute to the problem of elastic shape deformation by proposing an interactive alternative to Space-Time Constraints as efficient as purely geometric methods, yet offering a rich set of controls over the dynamics of interpolation: any pair of poses can be interpolated, and the frequency, amplitude, and damping of vibration modes can be intuitively and interactively edited.

1.4 Algorithm Overview

Our approach consists of a succession of a few simple steps, some performed offline before any user-interaction, and some performed at runtime to allow for interactive design of a dynamic shape interpolation between two poses.

- Pre-computation: Given a tetrahedron mesh in its rest shape, we compute the vibration modes W through classical Modal Analysis. Then we convert them into a basis $\widehat{W}$ in rotation-strain space (Section 2.2), capable of representing extremely large deformation.
- Runtime computation: Key frames $x(0), x(T)$ are also converted into rotation-strain space as $\widehat{x}(0), \widehat{x}(T)$ (Section 2.1). Their modal coordinates $z(0), z(T)$ are then computed through the relation $\widehat{W}z(t) = \widehat{x}(t)$. With (optional) user controls, we interpolate $z(0)$ and $z(T)$ according to the free vibration motion equation with optimal frequency and damping parameters (Section 2.3), and finally reconstruct each shape of the interpolation through Poisson reconstruction (Section 2.1).

2 Dynamic Shape Interpolation

We now elaborate on each individual component of our approach, the overall algorithm being given in Section 2.4.

2.1 Rotation-Strain Coordinates

To help us deal with very large deformation, we introduce *rotation-strain coordinates* based on polar decomposition of the deformation gradient of a pose. Given a shape vector x, its deformation gradient with respect to the rest shape r is computed as $m = Gx$ (one 3x3 matrix per tet), where G is the commonly-used discrete gradient operator corresponding to the continuous gradient operator ∇ with respect to r through linear finite elements [15]. For each tet T_i, we further decompose $m_i{=}(Gx)_i$ using polar decomposition [16] into the product of a rotation R_i and a symmetric tensor $\mathrm{Id}{+}S_i$ (where Id is the identity matrix, and S_i is the Biot strain tensor [16]). Similarly to [17], we finally apply the logarithm function to the rotation matrix, thus decomposing the deformation gradient m_i per tet into a pair $[\log(R_i), S_i]$. The array $\widehat{x} := \{\{\log(R_i)\}_i, \{S_i\}_i\}$ (where the index i goes through every tet) thus encodes the deformation gradient for the whole shape x, and forms what we will refer to as *rotation-strain coordinates*. We will denote by $\mathcal{T}(\cdot)$ the map between the Euclidean coordinates of shape x and the rotation-strain coordinates $\widehat{x}{:=}\mathcal{T}(x)$. Note that this map is non-linear, but trivial to compute through local polar decomposition of the deformation gradient. Note also that by definition, the rest shape r has null rotation-strain coordinates: $\widehat{r} = 0$.

Finally, one can recover the shape x from the rotation-strain coordinates $\widehat{x} = \{\{\log(R_i)\}_i, \{S_i\}_i\}$ through a pseudo-inverse map $\mathcal{T}^{-1}(\widehat{x})$, defined through exponentials of matrices (or simpler group difference maps) and a linear solve:

$$\boxed{\begin{array}{c}\mathcal{T}^{-1}(\widehat{x}) = \arg\min_{x} \sum_{\text{tet } T_i} \|(Gx)_i - [\exp(\log(R_i))(\mathrm{Id}+S_i)]\|^2 |T_i| \\ \text{subject to:} \quad \frac{\sum_{i=1}^{n} x_i |V_i|}{\sum_{i=1}^{n} |V_i|} = c,\end{array}} \tag{1}$$

where the position constraint is used to remove the translation invariance of our representation, $\|\cdot\|$ denotes the Frobenius norm of matrices, $|T_i|$ is the volume of tetrahedron T_i, and $|V_i| = \sum_{j\in\{k:x_i\in T_k\}} |T_j|/4$ denotes the volume associated with node x_i. The position c can, *e.g.*, be set to the linear interpolation of the two barycenters of boundary shapes, or set to follow a parabola if the object is supposed to be free-falling. Note that this shape recovery from our coordinates is essentially a 3D Poisson reconstruction, as $\widehat{x}$ derives from the deformation gradient field. By turning the constraints into (weighted) penalty terms added to the target function, this least-square system can be solved with a typical symmetric sparse linear system solver (we use UMFPACK [18] in our implementation), for which Cholesky factorization can further improve efficiency, since the coefficient matrix is determined only by the rest shape and the constraints and will not change during the interpolation.

2.2 Non-linear Vibration Modes

As we now detail, the rotation-strain space we defined is a particularly convenient way to extend linear modal analysis (that we review briefly first) and produce analytical expressions of very large vibrations.

2.2.1 Modal Analysis Overview

When considering small deformations around the rest shape, linear elasticity is a good alternative to the full treatment of nonlinear dynamics. Equations of motion are written in terms of the *displacement* field $u = x - r$ between a current shape x and the rest state r through:

$$Ku + D\dot{u} + M\ddot{u} = 0 \tag{2}$$

where K is the classical linear finite element stiffness matrix (the Hessian of the quadratic potential energy of the deformable body), while M is the lumped mass matrix (see [15] for details). Matrix D describes the commonly-used *Rayleigh damping*, defined as $D = \alpha_K K + \alpha_M M$, where α_K and α_M are damping parameters.

Further simplifications can be exploited through *Modal Analysis* [9][19], an approach that makes use of the solutions of the generalized eigenproblem:

$$Ky = \lambda^2 My, \tag{3}$$

which are vibration (eigen)modes W_i of (eigen)frequency λ_i representing the natural characteristic displacements that the elastic object can undergo. After assembling the modes in a *modal displacement matrix* $W = (W_1\, W_2 \ldots W_n)$ and the eigenvalues in a diagonal *modal frequency matrix* $\Lambda = \mathrm{diag}(\lambda_1^2, \lambda_2^2, \ldots, \lambda_n^2)$, one can verify that the modal displacement matrix W diagonalizes both stiffness and mass matrices:

$$W^t K W = \Lambda \qquad W^t M W = \mathrm{Id}. \tag{4}$$

Consequently, if we decompose a time-varying deformation $u(t)$ into its modal components through

$$u(t) = W z(t) \tag{5}$$

where $z(t)$ stores the *modal coordinates* (representing the "magnitudes" of all the eigenmodes), then Eq. (2) can be written as:

$$\boxed{\Lambda\, z + (\alpha_K \Lambda + \alpha_M \mathrm{Id})\, \dot{z} + \ddot{z} = 0.} \tag{6}$$

Notice that the dynamics of each eigenmode can now be computed independently since Λ is diagonal. Finally, the shape trajectory $x(t)$ can be recombined simply by linear superposition of each modal magnitude through $x(t) = r + W z(t)$ to get the resulting elastic motion. For efficiency, one may only keep the modes corresponding to low frequencies as this drastically reduces the amount of computations necessary to generate a realistic motion.

2.2.2 Non-linear Geometric Extension of Linear Modes

Modal analysis is particularly attractive as there is a *linear shape reconstruction map* between the modal magnitudes z and the current shape x as we just showed. Alas, this linearity is also its biggest limitation: such a fully linear treatment lacks rotation invariance (*i.e.*, a finite rotation induces a non-zero strain), leading to dramatic visual artifacts for large deformation. A simple remedy resides in the use of a corotational method in computational mechanics [20]: one can rewrite the displacement field per tet in a different, rotated frame (traditionally called "corotated (CR) frame") to obtain a smaller corotated displacement field u_{CR}. Modal warping [11] exploits a similar factoring out of the local rotation to obtain a linear relation between u_{CR} and z. Unfortunately, this treatment still requires finding a displacement u coherent with u_{CR}, and a history-dependent integration is then needed to account for the rotations in time. To circumvent this path-dependency issue, the actual history $z(\tau)$ for $\tau \in [0,t]$ is replaced by a quasi-static ramping of $z(t)$ to get a plausible $u(t)$. While only very approximative for arbitrary deformation, this method convincingly outperforms the fully linear modal analysis without dramatically increasing computational time [21]—but it is not directly applicable in our case since there is no straightforward map from the displacement field u to the modal coordinates z (See Section 2.5).

We propose instead another extension of linear modal analysis, based on geometric extrapolation. We keep the modal analysis setup *as is* and we only modify the *final shape reconstruction*. That is, we still consider Eq. (6) as the set of ODEs that the mode coordinates z satisfy; however, we change the reconstruction map from z to x to incorporate some geometric nonlinearity. To understand our approach, consider a *small* deformation x around the rest shape r, *i.e.*, $x = r + u$ with u being small. Remembering that we refer to ∇ as the spatial gradient with respect to r, we know from conventional linear elasticity that:

$$\begin{aligned}\nabla x = \text{Id} + \nabla u &= \text{Id} + \frac{\nabla u + (\nabla u)^t}{2} + \frac{\nabla u - (\nabla u)^t}{2} \\ &= \text{Id} + \epsilon(u) + \omega^*(u)\end{aligned} \tag{7}$$

where $\epsilon(u)$ is the *symmetric strain tensor* and $\omega^*(u)$ is an antisymmetric tensor representing a *small rotation*, similar to the local angular velocity of each element when u is regarded as velocity. We exploit this physical interpretation of the deformation gradient to derive a shape X from u through our rotation-strain coordinates:

$$X := \mathcal{T}^{-1} \begin{bmatrix} \omega^*(u) \\ \epsilon(u) \end{bmatrix} \tag{8}$$

This last equation defines our shape reconstruction map from u and the final shape X which obviously differs from the linear-elastic shape x. Notice that when u is small, linear elasticity, modal warping, and our approach all coincide, and $X \equiv x$. However, for very large deformation when the linear elasticity treatment breaks down, we employ our nonlinear shape reconstruction to get physically-plausible results. We stress that for large deformation, neither modal warping nor our rotation-strain extrapolation are physically "correct", but they both extend modal analysis with visually pleasing results. As we show next, our method has the advantage to provide a direct map between modal coordinates z and shape X, and visually more pleasing output when the deformation is large.

2.2.3 Linear Relationship in Rotation-Strain Space

Following the conventional modal analysis treatment, we can decompose the time-varying displacement $u(t)$ into time-varying modal components $z(t)$ through $u(t) = Wz(t)$. Noticing here that $\omega^*(W_i)$ and $\epsilon(W_i)$ are in fact the rate of change of the rotation-strain coordinates for a change in the displacement along W_i, we can assemble a matrix $\widehat{W} = \{\widehat{W_1}, \widehat{W_2}, \ldots, \widehat{W_n}\}$, with $\widehat{W_i} := \{\omega^*(W_i), \epsilon(W_i)\}$, that will satisfy:

$$\widehat{W} z(t) = \begin{pmatrix} \omega^*(u(t)) \\ \epsilon(u(t)) \end{pmatrix}. \tag{9}$$

With this linear relationship established, we can write our shape reconstruction map as a linear map in rotation-strain space:

$$\boxed{\widehat{X}(t) = \widehat{W} z(t).} \tag{10}$$

This expression, equivalent to Eq. (8) (with $\mathcal{T}$ applied to both sides), provides another geometric interpretation of our approximation: we in fact extrapolate modes to large displacements through an approach similar to "as-rigid-as-possible" shape interpolation which linearly interpolates the rotation and stretch parts as well, where the modal coordinate z_i plays the role of time t in [1]. Indeed, we take the infinitesimal rotation $\omega^*(W_i)\, dz_i$ and stretch $\text{Id} + \epsilon(W_i)\, dz_i$ induced by each mode i around the rest position, and extrapolate these deformations for a larger modal coordinate z_i using $\exp(\omega^*(W_i)\, z_i)$ as the local rotation and $\text{Id} + \epsilon(W_i)\, z_i$ as the stretch. While this extension from linear modal analysis does not guarantee exact physical behavior, it gives plausible shapes and motions at interactive rates (see Fig. 2). Importantly, because we now have a simple, time independent map between modal coordinates z and shape X, we can find a closed-form expression for the time evolution of the shape in rotation-strain space as we show next.

2.3 Analytical Vibration Magnitudes

Since the time evolution of modal coordinates z satisfies Eq. (6), closed-form solutions are easily derived. For each mode i, the modal coordinate z_i follows

$$\ddot{z}_i + (\alpha_K \lambda_i^2 + \alpha_M)\dot{z}_i + \lambda_i^2 z_i = 0, \tag{11}$$

an ODE that can be integrated *analytically*: each mode i vibrates exactly as a damped harmonic oscillator:

$$\boxed{z_i(t) = (P_i \cos(\omega_i t) + Q_i \sin(\omega_i t)) e^{-\alpha_i t},} \tag{12}$$

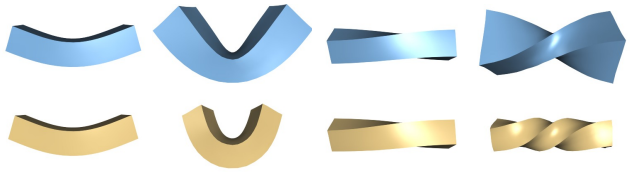


Fig. 2. Vibration of a rectangular box: 7^{th} mode (left two columns) and 9^{th} mode (right two columns). Compared with linear vibration mode (blue), our nonlinear extension (yellow) achieves significantly better visual results, especially under large deformation.

where the (angular) frequency is defined as $\omega_i = \sqrt{\lambda_i^2 - \alpha_i^2}$, the decay rate as $\alpha_i = (\alpha_K \lambda_i^2 + \alpha_M)/2$, and P_i and Q_i are two arbitrary scalar values depicting the initial amplitude and phase of mode i. (Note that we ignore the over- and critically-damped cases, as they rarely produce the vibration effects that we seek).

Boundary Conditions If the values of the i^{th} mode coordinate $z_i(t)$ are known at time $t=0$ and $t=T$, the values P_i and Q_i are then fully determined:

$$P_i = z_i(0), \quad Q_i = \frac{-z_i(0)\cos(\omega_i T) + z_i(T)e^{\alpha_i T}}{\sin(\omega_i T)}. \tag{13}$$

Adjustments to Bound Derivatives Fixing these two values, however, leaves no control over the time derivative $\dot{z}_i(0)$ of the coordinate at time 0:

$$\dot{z}_i(0) = Q_i \omega_i - \alpha_i z_i(0) \tag{14}$$

This derivative can in fact be quite large, potentially resulting in strong vibrations early on in the interpolation—note that if we control this initial derivative, the final time derivative (at time $t=T$) will also be small, as we use a *damped* oscillator on purpose with $\alpha_i \geq 0$ to achieve physically plausible energy dissipation over time. One can tweak ω_i and α_i in order to make $\dot{z}_i(0)$ as small as possible while minimizing the change of dynamics through solving:

$$\min_{\omega_i', \alpha_i' \geq 0} \gamma_z \dot{z}_i(0)^2 + \gamma_\omega(\omega_i' - \omega_i)^2 + \gamma_\alpha(\alpha_i' - \alpha_i)^2 \tag{15}$$

where $\gamma_z, \gamma_\omega, \gamma_\alpha$ are coefficients to weight the three terms (we use $\gamma_z = 0.5, \gamma_\omega = 0.25, \gamma_\alpha = 0.25$ in our implementation). A direct Levenberg-Marquart routine (**lmdif** in minpack) is used to efficiently solve this minimization for each mode.

2.4 Dynamic Shape Interpolation

We now have all the tools to define our dynamic shape interpolation. Given a rest shape r, we first perform the following precomputations:

- Assemble a stiffness matrix K, a lumped mass matrix M, and a deformation gradient matrix G using conventional linear finite elements (see, for instance, [15]).
- Perform the eigen-analysis of Eq. (3). To increase efficiency of further computations, keep only the leading m modes (*i.e.*, the eigenmodes corresponding the m smallest eigenfrequencies λ_i).
- Convert the non-zero m eigenmodes W_i to the rotation-strain space through $\omega^*(W_i)$ and $\epsilon(W_i)$, and assemble them into the matrix $\widehat{W}$ as explained in Section 2.2.3.

Now, for any given poses x_A and x_B sharing the same connectivity as r, we can compute and modify the dynamic shape interpolation at runtime by proceeding as follows:

- Using user-defined damping coefficients $\{\alpha_K, \alpha_M\}$, evaluate the vibration frequencies ω_i and decay rates α_i for each mode $i \leq m$ as described in Section 2.3.
- To have $x(0) = x_A$ and $x(T) = x_B$, decompose both poses into their rotation-strain coordinates as explained in Section 2.1, yielding $\widehat{x}(0)$ and $\widehat{x}(T)$ respectively.
- Using the pseudoinverse of $\widehat{W}$, obtain the modal coordinates at time $t = 0$ and $t = T$, *i.e.*, we compute $z(0)$ and $z(T)$ such that: $\widehat{W}z(0) = \widehat{x}(0)$ and $\widehat{W}z(T) = \widehat{x}(T)$, respectively.
- From this initial and final value of z, adjust the vibration frequency ω_i and damping α_i slightly (to ω_i' and α_i', respectively) to make sure that the interpolation is without obvious artifacts as detailed in Section 2.3.
- Eq. (12) then provides the analytic expression of $z(t)$ for every $t \in [0, T]$.
- Finally, recover the time-varying shape $X(t)$ defined in Eq. (10) by performing Poisson reconstructions as detailed in Eq. (1).

This approach is particularly efficient when the number m of modes is not too large (we use $m = 80$ or $m = 100$ in our examples). The only artifact we witnessed for small values of m is in the interpolation at $t=0$ and $t=T$: as the frequency domain is truncated for efficiency, not all specified shapes can be reached in the subspace spanned by the first m columns of $\widehat{W}$. Therefore, after projection through left multiplication of the pseudoinverse of $\widehat{W}$, there may be some non-zero residual $\rho_A = x_A - \mathcal{T}(\widehat{W}z(0))$ and $\rho_B = x_B - \mathcal{T}(\widehat{W}z(T))$. We found it sufficient simply to add their linear temporal interpolation to the reconstructed motion to remove any inaccuracies in the matching of intial and final poses. Alternatively, smooth splines or even Wiggly Splines could be used instead of this simple linear interpolation of residuals if the number of modes m used is so low that the residuals are significant. With our residual treatment, keyframes can be created by any algorithm or user interface: in fact, the poses used in our examples were created by various interactive mesh deformation algorithms such as Poisson mesh editing.

2.5 Comparison to Modal Warping

One may notice here that Modal Warping could be used to efficiently reconstruct the shape $\tilde{x}$ from z with the

following equation:

$$\tilde{x}(z) = r + \int_0^1 \exp(\Omega z) W z = r + \tilde{R}(\Omega z) W z \qquad (16)$$

where Ω maps z to $\log(R_i)$ for each vertex i by averaging over all the rotational part of $\hat{W}$ in each tet adjacent to the vertex. (More details about the form of $\tilde{R}$ can be found in [11].) Recovering z from x can be achieved through:

$$z := \arg\min_z \sum_{i=1}^{n} \|x_i - \tilde{x}_i(z)\|^2 |V_i|, \qquad (17)$$

or

$$z := \arg\min_z \sum_{\text{tet } T_i} \|(Gx)_i - (G\tilde{x}(z))_i\|^2 |T_i|. \qquad (18)$$

However, the Hessian of either of these two nonlinear optimizations is a dense matrix with a dimensionality equal to the number of modes used. Consequently, our tests showed that examples with highly deformed keyframes for which a small number of modes is used either converge to a visually-displeasing local minimum when using Eq. (17), or result in large residual (see Fig. 3). In fact, only when we initialize the minimization with the result that our rotation-strain space method provides can we get satisfying results. Although slightly more time consuming because of the Poisson reconstruction, our approach is more appropriate to robustly handle large meshes and very complex deformation: the ramping used in MW generates an error that grows with the amplitude of deformation, altering local velocities significantly, and thus distorting the shape greatly.

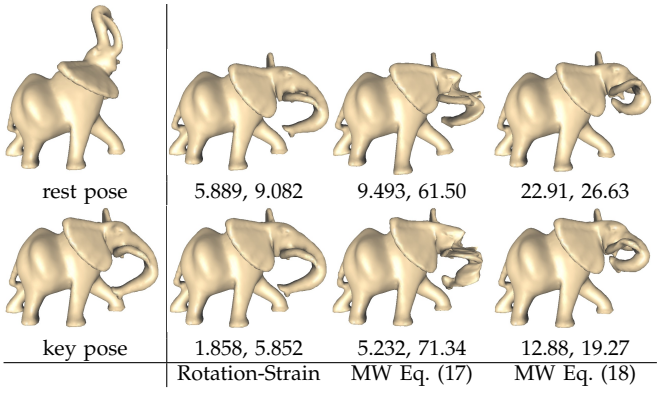


Fig. 3. Robustness compared to Modal Warping: Using the same 50 and 100 leading modes, our method can reconstruct the keyframe of the elephant model well without large residual. However, even if we use our result for the initialization of their optimization, Modal Warping leads to large errors. To quantize the error, we adopt the measurements of Eq. (17) and Eq. (18): The first number below the image measures the error in vertex position, and the second number measures the error in deformation gradient.

3 User Controls

Defining initial and final shapes is enough to produce dynamic shape interpolations through the algorithm we introduced above. However, we can allow the user to control many aspects of the dynamic interpolation with realtime feedback as we now review.

Frequency and Damping Control The eigenfrequencies stored in Λ can be edited to adjust the vibration period of each mode during the interpolation. For example, setting all the values of Λ close to zero means that each vibration frequency is very low, and the resulting interpolation will contain few vibration periods. On the contrary, one can increase the values of Λ to decrease their respective periods, rendering the interpolation more jiggly. Modifying the dynamic system by non-uniform scaling of K and M is feasible, however it would require solving a new eigenproblem—thus 1 to 10 seconds of additional computation. A global scaling of the stiffness (or mass matrix) can be easily achieved by rescaling all the frequencies by the square root of the scale (or its inverse, resp.).

The user can specify different damping values α_K and α_M to adjust the resulting dynamics of the interpolation. As expected, a large damping will make the vibrations settle down more quickly (Fig. 4)—and conversely. Further control, at the price of deviating a bit from the usual modal analysis framework, can be achieved by changing the various modal damping coefficients α_i; this alternative will help dissipate each modes more or less depending on the desired cinematic effects. We found that introducing a coefficient $\mu_i \in [-1, 1]$ per mode and substituting the damping α_i of vibration mode i by $\lambda_i(\mu_i + 1)/2$ is a particularly intuitive way to adjust the motion. Scaling with a coefficient $\eta_i \in [-1, 1]$ was also found to work well to adjust the frequencies λ_i^2 to $\lambda_i^2 \exp(\eta_i/(1-|\eta_i|))$. Based on these control variables, we developed a simplified interface, where a spline is used to edit the various vibration frequencies and damping. We found it particularly useful to edit the motion (see Fig. 5 and Fig. 6).

To achieve direct (vibration-free) shape interpolation, we can simply set both η_i and μ_i close to -1 for all modes, thus eliminating oscillations as shown in Fig. 12(a). Conversely, to enhance vibration effects, we can increase the frequencies of low frequency modes, which will result in noticeably more jiggly behavior (Fig. 4). High frequency modes can also provide subtle motion details as demonstrated in Fig. 7.

Position and Orientation Constraints Position and orientation constraints can be added during Poisson reconstruction. They can be interpolated through shape matching technique or based on keyframes, and then applied to the Poisson reconstruction for each frame. For instance, Fig. 8 shows an example where position constraints are used to fix the legs of the elephant throughout shape interpolation. Better result for position constraints could be achieved by modifying K and M to

Fig. 4. Various damping effects: we set a non-zero damping coefficient μ to make the animation settle down (bottom row; only the last three frames are shown here). Compared with the damping-free motion (top row), and similar initial velocities, the energy decay differs widely.

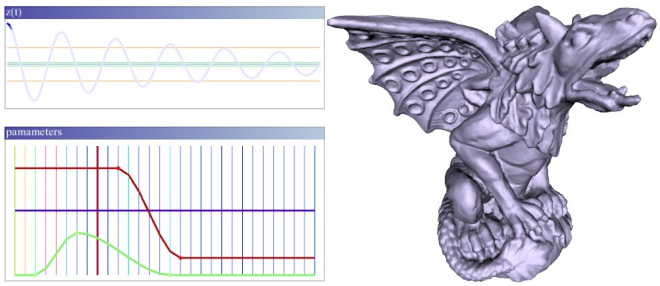


Fig. 5. User interface: by editing the curves (green for damping control μ, red for frequency control η, and the rest for γ's) in the parameter window (bottom left), the user can intuitively specify μ, η and γ for each mode, and monitor $z(t)$ of the selected mode in the other window (top left). Optimization of the parameters for each mode can be done efficiently, aiding design through visual feedback at interactive rate.

make low frequency modes correspond to low frequency modes of the constrained vibration, however the computational cost would render the process non-interactive.

Removing Self-Intersections and Adding Keyframes Our results can contain self-intersections if shapes x_A and x_B are quite different. Moreover, the user may not be satisfied with what she obtained using a direct application of our technique. To deal with these two issues and allow for greater editing capabilities, we let the user provide keyframes to alter the interpolation interactively. Given a set of k keyframes $\{\underline{X}(t_i), i = 1, 2, \ldots, k\}$ where the t_i's represent the times at which the keyframes need to be reached, we first evaluate the adjustments in rotation and strain needed for our interpolation to meet the i^{th} keyframe through

$$\widehat{\Delta}_i = \mathcal{T}(\underline{X}(t_i)) - \mathcal{T}(X(t_i)). \tag{19}$$

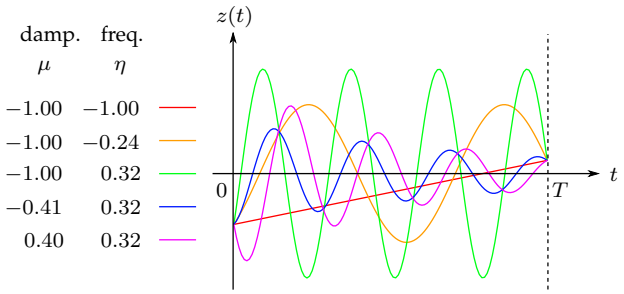


Fig. 6. Examples of various vibrating curves with different damping and frequency parameters, all satisfying the same boundary constraints.

We then smoothly blend the keyframes to the shape interpolation using a modified shape X^* defined as

$$\widehat{X}^*(t) = \widehat{X}(t) + \sum_{i=1}^{i=k} \phi_i(t)\widehat{\Delta}_i, \tag{20}$$

where $\phi_i(t)$ is a cubic spline with support in $[t_{i-1}, t_{i+1}]$. To ensure smoothness of the modified interpolation, we set the spline derivatives at t_{i-1}, t_i and t_{i+1} to 0. This simple procedure offers additional control over the results as Fig. 9 demonstrates. As shown in Fig. 10, interpolating multiple key frames with Wiggly Splines is also straightforward. After converting key frames into modal coordinates, we specify internal nodes for the vibration curve of each mode through Wiggly Splines, which makes the interpolation curve "as physical as possible" for that mode. Note that velocity constraints could also be incorporated through conversion of velocity into the rotation-strain space. Thus, our approach provides a simple computation of nonlinear modes a la Wiggly Splines, instead of going through linear PCA of large deformation simulations.

Quasi-Statics vs. Dynamics While our method provides an intuitive way to design dynamic interpolation between shapes, the user may want to *start from an existing, quasi-static shape interpolation* $\underline{X}(t)$ (obtained through [22], [23], [3], or [6] for instance) and *add elastic oscillations* to this prescribed shape interpolation. This is achieved by computing the transform $\widehat{\underline{X}(t)} := \mathcal{T}(\underline{X}(t))$ of the input shape sequence (we will write its coordinates as: $\hat{\underline{X}} = \{\log R(\underline{X})_i, S(\underline{X})_i\}$). Now the only change in

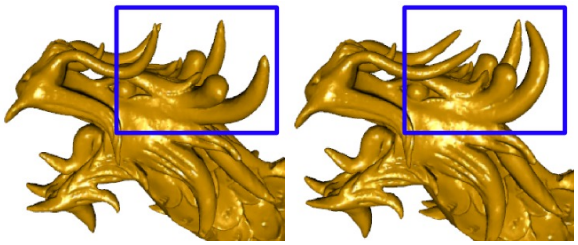


Fig. 7. Only exciting high frequency modes can make the geometric details (e.g., horn and whiskers) quiver while keeping the low frequencies (global motion) smooth.

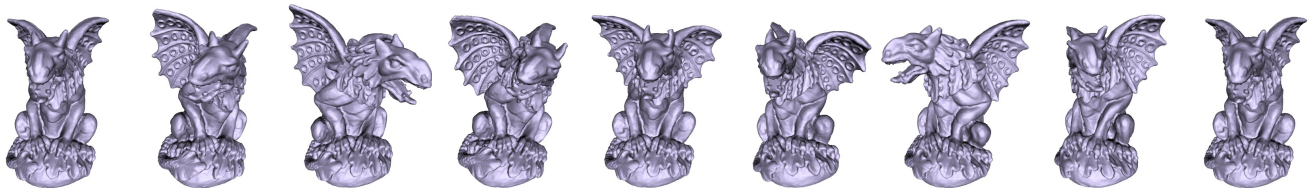


Fig. 10. Our rotation-strain formulation allows us to use Wiggly Splines to interpolate between four key frames (given at time $0, 0.25, 0.75$ and 1 s.), where both the first and the last pose are the rest state.

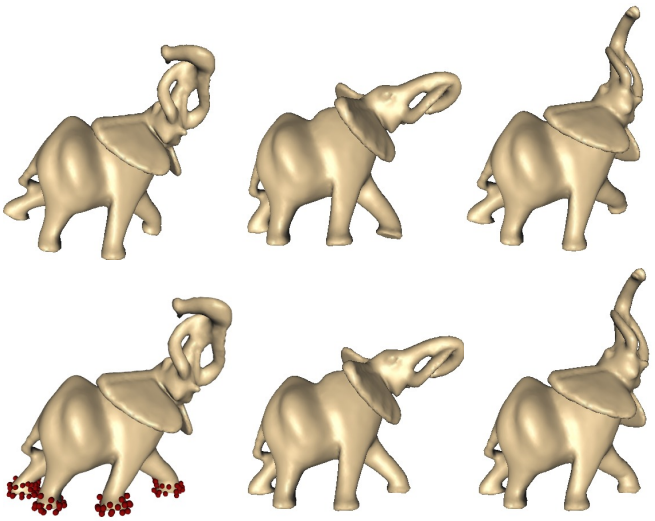


Fig. 8. The feet of the elephant will not stay on the ground if we only use a barycenter constraint in Poisson reconstruction (top row). Position constraints (red dots) can instead be added to control placement of the feet.

our dynamic interpolation procedure is to use $\widehat{X}(t) = (1-\gamma)\ \underline{\widehat{X}} + \gamma \widehat{W} z(t)$ instead of just $\widehat{W} z(t)$. With these minor modifications, the main motion (given as a quasi-static motion that interpolates x_A and x_B) is enriched with secondary deformations (based on our extended modal analysis) that can be controlled with intuitive parameters.

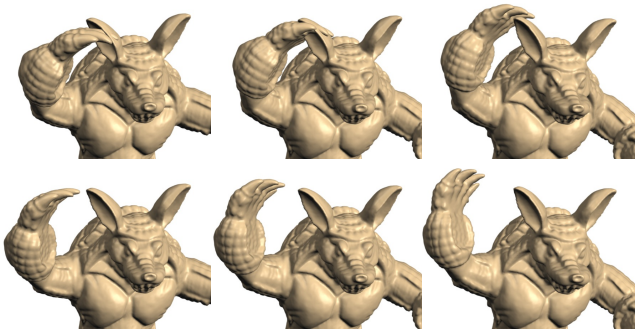


Fig. 9. Self-intersections (top row) can be resolved by adjusting a frame (top, center) to be intersection free (bottom), and inserting this adjusted shape as a key frame along the interpolation. Self-intersections in nearby frames will then be avoided, while the vibration effects in the whole interpolation are kept nearly intact.

Other Vibration Modes We used the vibration modes resulting from modal analysis in our exposition, but any other process can be used as long as a series of "characteristic" displacements $\{W_i\}$ are produced. While the eigenmodes we use can be tweaked by changing the Lamé coefficients of the elastic potential (or even by using spatially-varying material coefficients), we also tried using thin-shell elastic model [24] as shown in Fig. 11. After calculating the linear vibration modes and corresponding frequency of the triangle mesh, we evaluate the deformation gradient by adding a fourth node for each triangle as proposed in [22]. As expected, the modes visually correspond to whatever model we selected, and the nonlinear extension presented in Section 2.2.2 can be performed *as is* independently of what was used to generate the basic eigenmodes.

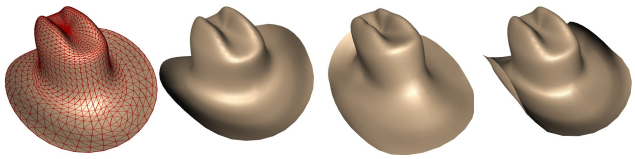


Fig. 11. We can interpolate between triangle meshes as well, using a deformation model governed by thin-shell elastic energy. The rest shape is (a). The key frames in this example are (a) and (d). (b) and (c) are intermediate frames in the animation sequence obtained by our method.

4 Implementation and Performances

4.1 Handling Excessive Twists

When the start or end shape is extremely twisted, the $\log(R)$ of adjacent elements computed according to Section 2.1 may have opposite signs. During the interpolation, such elements will rotate in almost opposite directions, leading to visual artifacts. To address this issue and project such poses to their correct rotation-strain coordinates, we traverse the elements in the mesh with a breadth-first search starting from a seed tetrahedral element after extracting the unit rotation axis from $\log(R)$ with a positive angle in the interval of $[0, \pi]$). When we find an element with its unit rotation axis

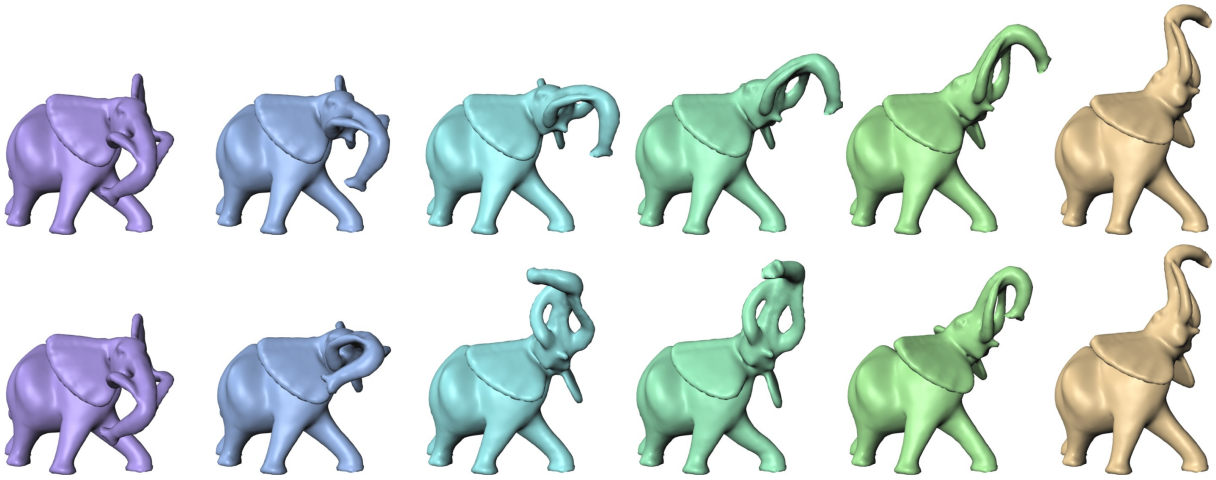


Fig. 12. Our shape interpolation can interactively provide a dynamic motion between two poses. Here, the interpolation is demonstrated on an elephant model, the first pose being with the trunk down (leftmost) and the final pose (the rest state) being with the trunk up in the air (rightmost). Although many dynamical coefficients can be interactively edited, we show a direct vibration-free interpolation (top row, $\mu = -1$ and $\eta = -1$), to compare it to the effect of adding low frequencies to the motion (bottom row): the trunk and ears now dynamically deform throughout the pose interpolation.

differing greatly from those of its (traversed) neighbors (dot product of the unit rotation axes < -0.5), we adjust the rotation angle by adding or subtracting the smallest times of 2π to solve this problem. With this added treatment, our method can successfully handle severely deformed shapes involving local rotations larger than 2π.

4.2 Initializing Levenberg-Marquart Optimization

We set the initial value to $\omega_i' = \omega_i, \alpha_i' = \alpha_i$ for the Levenberg-Marquart routine which is used for the optimization problem introduced in Section 2.3. Although the routine may be trapped in local minima, in most cases, the results turn out satisfactory. In rare cases, it fails to converge or produces very poor result (with unreasonably large initial velocity). Simply trying several (less than 5 in our experience) different random initial values around ω_i and α_i solves this problem.

4.3 Initial Velocity Reduction

If the first frame x_A is very close to the rest shape and the end frame x_B is highly deformed, a large gain of potential energy needs to happen during the process. Under such circumstances, large damping often leads to unnatural behavior because of the unavoidably large initial velocity. To avoid high initial kinetic energy, we can choose a shape x_r between x_A and x_B as the "rest shape" without doing any additional offline precomputations by simply offsetting the rotation-strain coordinates through $\widehat{x}_r = (1-\beta)\widehat{x}_A + \beta\widehat{x}_B$.

4.4 Performance

Our approach, along with the various user controls described above, was implemented on a PC with 2.0GHz Intel Xeon CPU and an NVidia GeForce 8600GT graphics card. Performance statistics for the various examples shown in our paper can be found in Table 1. To maintain interactive performance even for huge meshes, we can also embed the detailed triangle mesh into a simplified tetrahedron mesh. To generate such a tetrahedron mesh, we first simplify the triangle mesh (e.g., the gargoyle model is simplified to 2000 vertices), then offset the vertices of the simplified mesh along the normal to enclose the input mesh. Finally, we use NETGEN [25] to tetrahedralize the simplified surface mesh. The triangle meshes used as key frames and outputs are easily interpolated from the deformed tetrahedron mesh. A dense tessellation may be required to capture high frequency vibration modes; thankfully, we can deal with 20K tet meshes still interactively, which proved sufficient for the design of all the dynamical effects in this paper (see Fig. 1).

5 Conclusion

We presented a shape interpolation algorithm built on decoupled vibration modes extrapolated in a rotation-strain space to generate artifact-free large deformation. A simple mapping between shape space and modal coordinates enables dynamic morphing between a pair of shapes, while offering interactive control over the various parameters of the motion. Moreover, our method can compute the rotation-strain space coordinates of added key frames efficiently, leading to truly interactive design of motion.

Limitations of our method include the sacrifice of physical accuracy for speed and the absence of check for plausibility of the user's choice of parameters, which can lead to visual results far from resembling the behavior of any existing material. For fast-moving skeleton-driven animation, inertial forces may need to be taken into

model	mesh vertices	tet nodes	tet elements	modes	eigen.	opt.	$\widehat{X}(t) = \widehat{W}z(t)$	$\mathcal{T}^{-1}(\widehat{X})$	fps
box	440	440	1337	50	1s	6ms	2ms	8ms	70
hat	1607	4771	3164	50	7s	6ms	9ms	25ms	23
elephant	9k	1636	5442	50	6s	6ms	6ms	33ms	20
armadillo	17k	1628	5249	100	7s	12ms	12ms	30ms	18
dragon	10k	2678	9237	100	11s	12ms	22ms	55ms	10
gargoyle	10k	2957	11054	100	12s	12ms	24ms	62ms	9
raptor	20k	1251	3677	50	4s	6ms	5ms	24ms	23

TABLE 1
Performance statistics for the models presented in this paper; *eigen.* indicates the time it took to perform eigenanalysis using Matlab, while *opt.* gives timings of the Levenberg-Markart optimization.

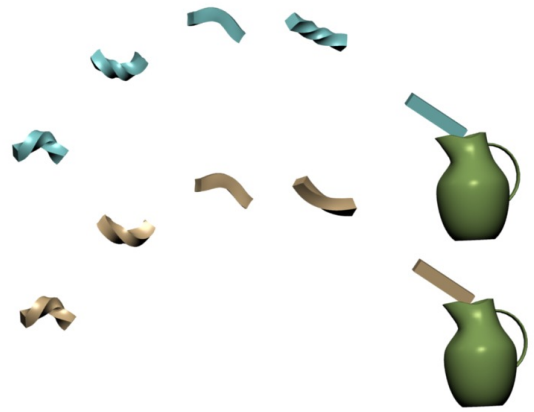


Fig. 13. The motion of a rubber stick (straight at rest) thrown into a jar can be interactively edited to achieve various cinematic effects. In this example, the bottom row uses a larger damping coefficient for the mode leading to a twist-like deformation.

account in the form of modal forces. It would also be interesting to explore the extension of the motion subspace through modal derivatives [26].

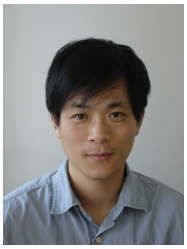
Acknowledgments: We would like to thank the reviewers for their valuable comments. This research was partially supported by NSFC (No. 60703039, 60933007), China 973 Program (No. 2009CB320801), the Fundamental Research Funds for the Central Universities (No. 2009QNA5023), and Yiying Tong, Mathieu Desbrun are supported by NSF grants CCF-0811313 & 0811373 & 1011944, CMMI-0757106 & 0757123 and IIS-0953096, as well as support from Pixar Animation Studios.

REFERENCES

[1] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," in *Proceedings of ACM SIGGRAPH*, 2000, pp. 157–164.
[2] S. Kircher and M. Garland, "Free-form motion processing," *ACM Trans. Graph.*, vol. 27, no. 2, pp. 1–13, 2008.
[3] M. Kilian, N. J. Mitra, and H. Pottmann, "Geometric modeling in shape space," *ACM Trans. Graphics*, vol. 26, no. 3, 2007.
[4] A. Witkin and M. Kass, "Spacetime constraints," in *Proceedings of ACM SIGGRAPH*, 1988, pp. 159–168.
[5] J. Popović, S. M. Seitz, M. Erdmann, Z. Popović, and A. Witkin, "Interactive manipulation of rigid body simulations," in *Proceedings of ACM SIGGRAPH*, 2000, pp. 209–217.
[6] B. Adams, M. Ovsjanikov, M. Wand, H.-P. Seidel, and L. J. Guibas, "Meshless modeling of deformable shapes and their motion," in *Symposium on Computer Animation*, 2008.
[7] J. Barbič and J. Popović, "Real-time control of physically based simulations using gentle forces," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–10, 2008.
[8] J. Barbič, M. da Silva, and J. Popović, "Deformable object animation using reduced optimal control," *ACM Trans. Graph.*, vol. 28, no. 3, 2009.
[9] A. Pentland and J. Williams, "Good vibrations: modal dynamics for graphics and animation," in *Proceedings of ACM SIGGRAPH*, 1989, pp. 215–222.
[10] D. L. James and D. K. Pai, "DyRT: dynamic response textures for real time deformation simulation with graphics hardware," in *Proceedings of ACM SIGGRAPH*, 2002, pp. 582–585.
[11] M. G. Choi and H.-S. Ko, "Modal warping: Real-time simulation of large rotational deformation and manipulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 1, pp. 91–101, 2005.
[12] W.-W. Feng, B.-U. Kim, and Y. Yu, "Real-time data driven deformation using kernel canonical correlation analysis," in *ACM Trans. Graph.*, 2008, pp. 1–9.
[13] T. Popa, Q. Zhou, D. Bradley, V. Kraevoy, H. Fu, A. Sheffer, and W. Heidrich, "Wrinkling captured garments using space-time data-driven deformation," *Computer Graphics Forum (Proc. Eurographics)*, vol. 28, no. 2, 2009.
[14] M. Kass and J. Anderson, "Animating oscillatory motion with overlap: wiggly splines," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–8, 2008.
[15] K. J. Bathe, *Finite Element Procedures in Engineering Analysis*. Englewood Cliffs, N. J.: Prentice Hall, 1995.
[16] A. Bertram, *Elasticity and Plasticity of Large Deformations: An Introduction*. Springer, 2005.
[17] K. G. Der, R. W. Sumner, and J. Popović, "Inverse kinematics for reduced deformable models," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1174–1179, 2006.
[18] T. A. Davis, "Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, vol. 30, no. 2, pp. 196–199, 2004.
[19] K. Hauser, C. Shen, and J. F. O'Brien, "Interactive deformations using modal analysis with constraints," in *Proceedings of Graphics Interface*, Jun. 2003, pp. 247—256.
[20] B. F. de Veubeke, "The dynamics of flexible bodies," *International Journal of Engineering Science*, vol. 14, pp. 895–913, 1976.
[21] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, December 2006.
[22] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 399–405, 2004.
[23] D. Xu, H. Zhang, Q. Wang, and H. Bao, "Poisson shape interpolation," in *Proceedings of the ACM Symposium on Solid and Physical Modeling*, 2005, pp. 267–274.
[24] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder, "Discrete shells," in *Symposium on Computer animation*, 2003, pp. 62–67.
[25] J. Schöberl, "Netgen - an advancing front 2d/3d-mesh generator based on abstract rules," *Comput. Visual. Sci*, pp. 1:41–52, 1997.
[26] J. Barbič and D. L. James, "Real-time subspace integration for St.

Venant-Kirchhoff deformable models," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 982–990, Aug. 2005.
[27] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1134–1141, 2005.
[28] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler, "Stable real-time deformations," in *Symposium on Computer animation*, 2002, pp. 49–54.
[29] Z. Popović and A. Witkin, "Physically based motion transformation," in *Proceedings of ACM SIGGRAPH*, 1999, pp. 11–20.

Jin Huang is an associated professor in the state key laboratory of CAD&CG at Zhejiang University, P.R.China. He received his PhD. degree in Computer Science Department from Zhejiang University in 2007 with Excellent Doctoral Dissertation Award of CCF (China Computer Federation). His research interests include geometry processing and physically-based simulation. He has served as reviewer for ACM SIGGRAPH, EuroGraphics, Pacific Graphics, TVCG etc.



Yiying Tong is an Assistant Professor at Michigan State University. Prior to joining MSU, He worked as a postdoctoral scholar at Caltech. He received his Ph.D. degree from University of Southern California in 2004. His research interests include discrete geometric modeling, physically-based simulation/animation, and discrete differential geometry. He received the U.S. National Science Foundation (NSF) Career Award in 2010 . His list of publications is available at www.cse.msu.edu/~ytong.

Kun Zhou is a Cheung Kong Distinguished Professor in the Computer Science Department of Zhejiang University, and a member of the State Key Lab of CAD&CG, where he leads the Graphics and Parallel Systems Group. Prior to joining Zhejiang University in 2008, Dr. Zhou was a Leader Researcher of the Internet Graphics Group at Microsoft Research Asia. He received his BS degree and PhD degree in computer science from Zhejiang University in 1997 and 2002, respectively. His research interests include shape modeling/editing, texture mapping/synthesis, real-time rendering and GPU parallel computing.

Hujun Bao received the BS and PhD degrees in applied mathematics from Zhejiang University in 1987 and 1993, respectively. Currently, he is a Professor and the director of State Key Laboratory of CAD&CG at Zhejiang University. His main research interest is computer graphics and computer vision, including realtime rendering technique, geometry computing, virtual reality and 3D reconstruction.

Mathieu Desbrun is a Professor at the California Institute of Technology (Caltech). After receiving his Ph.D. from the National Polytechnic Institute of Grenoble (INPG), he spent a year as a post-doctoral researcher at Caltech before joining the Computer Science faculty at the University of Southern California from 2000 to 2004. He now leads the Applied Geometry lab at Caltech, focusing on discrete differential modeling and a wide spectrum of applications. He is also the director of the Computing & Mathematical Sciences department within the Information Science and Technology initiative at Caltech.