Interactive Shape Interpolation through Controllable Dynamic Deformation

Jin Huang* Yiying Tong† Kun Zhou* Hujun Bao* Mathieu Desbrun‡

*Zhejiang University †Michigan State University ‡Caltech

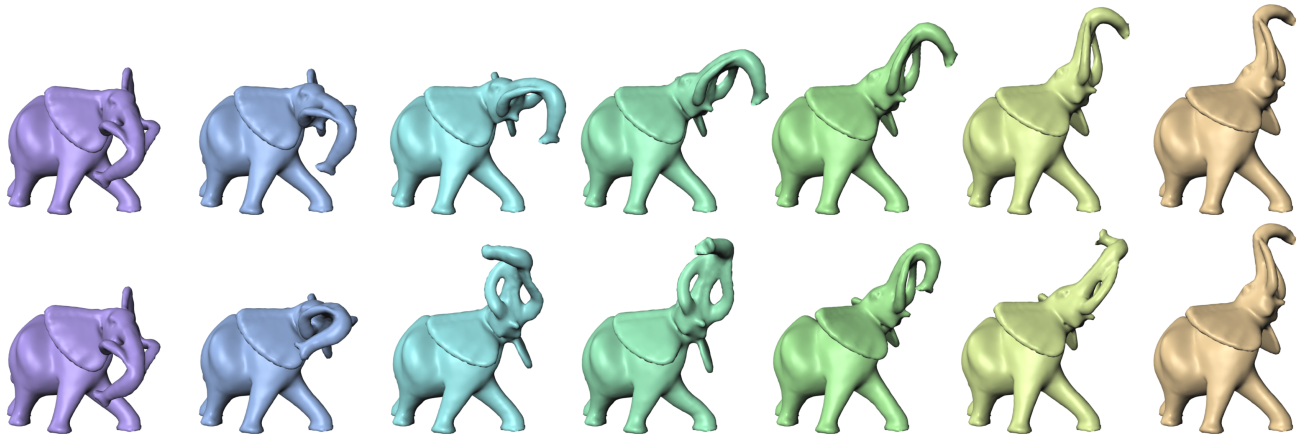

Figure 1: *Our shape interpolation can interactively provide a dynamic motion between two poses. Here, the interpolation is demonstrated on an elephant model, the first pose being with the trunk down (leftmost) and the final pose (the rest state) being with the trunk up in the air (rightmost). Although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

Abstract

In this paper, we introduce an interactive approach to generate physically-based shape interpolation between two poses. We combine an extension of modal analysis to large deformation and a Poisson reconstruction from deformation gradient to offer physically-plausible dynamics at interactive rates. 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.

Keywords: Deformation Gradient, Shape Interpolation, Space-Time Constraints

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 fastidious.

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:

$$\begin{cases} x(0) = x_A \\ x(T) = x_B, \end{cases}$$

while offering intuitive and interactive control over the dynamics throughout the interpolation sequence.

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 [Alexa et al. 2000]. 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 [Kilian et al. 2007].

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 [Witkin and Kass 1988; Popović et al. 2000]. 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 [Adams et al. 2008] 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[Barbič and Popović 2008], but it did not address the challenging problem of getting the initial trajectory connecting the input key frames at interactive rates.

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 [Pentland and Williams 1989; James and Pai 2002], 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 recently proposed to gracefully extend the original approach to large deformations [Choi and Ko 2005], 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 [Feng et al. 2008; Popa et al. 2009] 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* [Kass and Anderson 2008], 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 *large elastic deformations* akin to those in Modal Warping. However, our approach further allows a simple map between the shape change in time, and analytic solutions of the time evolution of modes. As a result, we 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, while the frequency, amplitude, and damping of vibration modes can be intuitively and interactively edited.

1.4 Algorithm Overview

Our algorithm is composed of a pre-computation part and a runtime computation part.

- 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 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 the shape through Poisson reconstruction (Section 2.1).

2 Dynamic Shape Interpolation

Our approach consists in 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. This section elaborates on each individual component one by one, before presenting the overall algorithm in Section 2.4.

2.1 Rotation-Strain Coordinates

To help us deal with 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 [Bathe 1995]. For each tet T_i, we further decompose $m_i = (Gx)_i$ using polar decomposition [Bertram 2005] 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 [Bertram 2005]). Similarly to [Der et al. 2006], 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 abusively refer to as *rotation-strain coordinates* for simplicity. 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 simple 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 maps $\mathcal{T}^{-1}(\widehat{x})$, defined through exponentials of matrices and a simple linear solve:

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

where the position constraint is used to remove the translation invariance of our representation, and $\|\cdot\|$ denotes the Frobenius norm of matrices. 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. Simply adding the (rescaled) norm of the constraint as a penalty term to the target function, we solve the unconstrained least square problem with the package UMFPACK [Davis 2004].

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 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 [Bathe 1995] for details). Matrix D describes the commonly-used Raleigh *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* [Pentland and Williams 1989; Hauser et al. 2003], 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* $\Lambda = \text{diag}(\lambda_1^2, \lambda_2^2, \ldots, \lambda_n^2)$, one can verify that the modal displacement matrix W diagonalizes both stiffness and mass matrices:

$$W^t K W = \Lambda \qquad W^t M W = \text{Id}.$$

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

$$u(t) = W z(t)$$

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

$$\boxed{\Lambda\, z + (\alpha_K \Lambda + \alpha_M \text{Id})\, \dot{z} + \ddot{z} = 0,} \tag{4}$$

Notice that the dynamics of each eigenmode can now be computed independently since Λ is diagonal. Finally, the shape trajectory $x(t)$ can easily be recombined 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 [de Veubeke 1976]: 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 [Choi and Ko 2005] exploits a similar factoring out of the local rotation to now have 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 increasing computational time dramatically [Nealen et al. 2006]—but it is not appropriate for our purposes since there is no longer a simple map from the displacement field u to the modal coordinates z independent of time.

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. (4) 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:

$$\nabla x = \text{Id} + \nabla u = \text{Id} + \frac{\nabla u + (\nabla u)^t}{2} + \frac{\nabla u - (\nabla u)^t}{2} = \text{Id} + \epsilon(u) + \omega^*(u),$$

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:

$$X := \mathcal{T}^{-1} \begin{bmatrix} \omega^*(u) \\ \epsilon(u) \end{bmatrix} \tag{5}$$

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 coincides, and $X \equiv x$. However, for 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 shown next, our method has the distinctive advantage to provide a direct map between modal coordinates z and shape X.

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 $u(t) = W z(t)$. 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:

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

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

$$\boxed{\widehat{X}(t) = \widehat{W} z(t).} \tag{6}$$

This expression, equivalent to Eq. (5) (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, where the modal coordinate z_i plays the role of time t

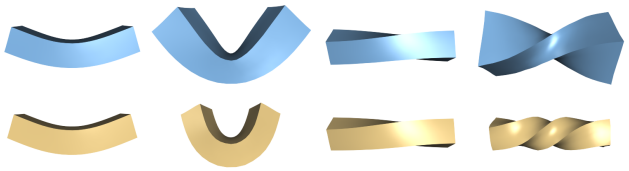

Figure 2: *Vibration of a rectangular box: 7^{th} mode (left two columns) and 9^{th} mode (right two columns). Compared with linear vibration mode (blue), our nonlinear extension (yellow) achieves significantly better visual results, especially under large deformation.*

in [Alexa et al. 2000]. Indeed, we take the infinitesimal rotation $\omega^*(W_i)\,dz_i$ and stretch $(\mathrm{Id}+\epsilon(W_i)\,dz_i)$ 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 $\mathrm{Id}+\epsilon(W_i)\,z_i$ 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 Figure 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. (4), closed-form, analytical solutions are easily derived. For each mode i, the modal coordinate z_i follows

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

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

$$\boxed{z_i(t) = (P_i \cos(\omega_i t) + Q_i \sin(\omega_i t))e^{-\alpha_i t},} \qquad (7)$$

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 do not 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:

$$P_i = z_i(0), \quad Q_i = \frac{-z_i(0)\cos(\omega_i T) + z_i(T)e^{\alpha_i T}}{\sin(\omega_i T)}. \qquad (8)$$

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:

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

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:

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

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 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, [Bathe 1995]).
- 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. (7) 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. (6) by performing Poisson reconstructions as detailed in Eq. (1).

This approach is particularly efficient when the number m of modes is small (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 in practice to simply 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.

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 Editing 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,

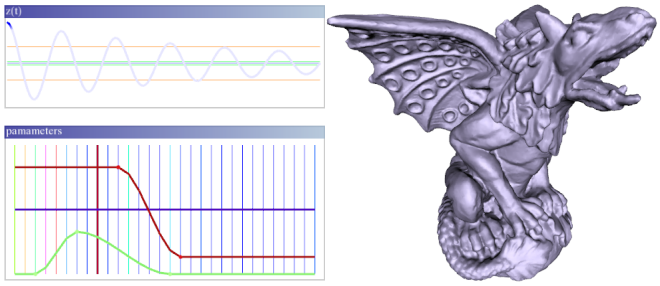

Figure 3: *User interface: by editing the curves (green for damping control μ, red for frequency control η, and the rest for γ's) in the parameter window (bottom lef), the user can intuitively specify μ, η and γ for each mode, and monitor $z(t)$ of the selected mode in the other window (top left). Optimization of the parameters for each mode can be done efficiently, enabling visual feedback at interactive rate.*

Figure 4: *Various damping effects: we set some non-zero damping coefficient μ to make the animation calm down gradually (the bottom row). Only last three frames were shown here. Compared with the damping-free motion (the top row), although the initial velcoities are similar, the energy decay differs widely. Please refer to the accompanying video for the complete interpolation.*

rendering the interpolation more jiggly. A simplified frequency editor, where a spline is used to edit the various vibration frequencies, was a particularly useful editing tool when adjusting results (see Figure 3). Modifying the dynamic system by non-uniform scaling of K and M is feasible, however it would require solving a new eigen problem—thus 1 to 10 seconds of additional computation.

Damping Control 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 (Figure 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 trick will help dissipating 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 $\sqrt{\lambda_i}(\mu_i + 1)/2$ is particularly intuitive 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 $\lambda_i^2 \exp(\eta_i/(1 - |\eta_i|))$ (see Figure 5).

To achieve direct (vibration-free) shape interpolation, we can simply set both μ_i and η_i close to -1 for all modes, thus eliminating

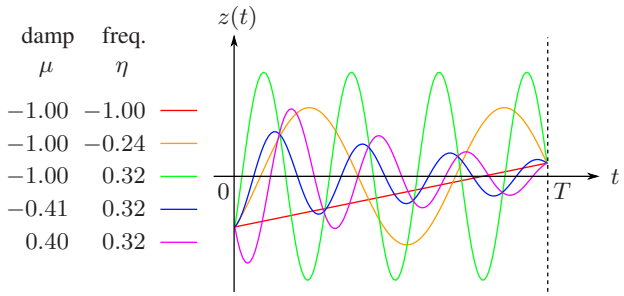

Figure 5: *Various vibrating curves satisfying the same boundary constraints.*

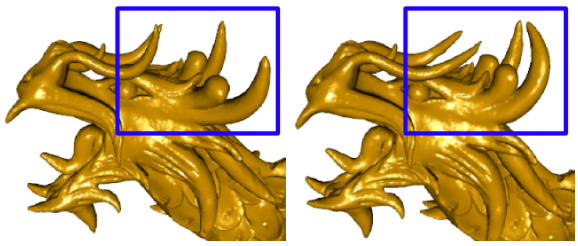

Figure 6: *Only exciting high frequency modes can make the geometric details (e.g., horn and beard) quiver while keeping the low frequencies (global motion) smooth.*

oscillations as shown in Figure 1(a). Conversely, to enhance vibration effects, we can increase the frequencies of low frequency modes, which will result in noticeably more jiggly behavior (Figure 4). High frequency modes can also provide subtle motion details as demonstrated in Figure 6.

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, Figure 7 shows an example where position constraints are used to fix the legs of the elephant throughout shape interpolation. Better result for position constraints can be achieved by modifying K and M to 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 too 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

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

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

$$\widehat{X}^*(t) = \widehat{X}(t) + \sum_{i=1}^{i=k} \phi_i(t)\widehat{\Delta}_i, \qquad (9)$$

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 deriva-

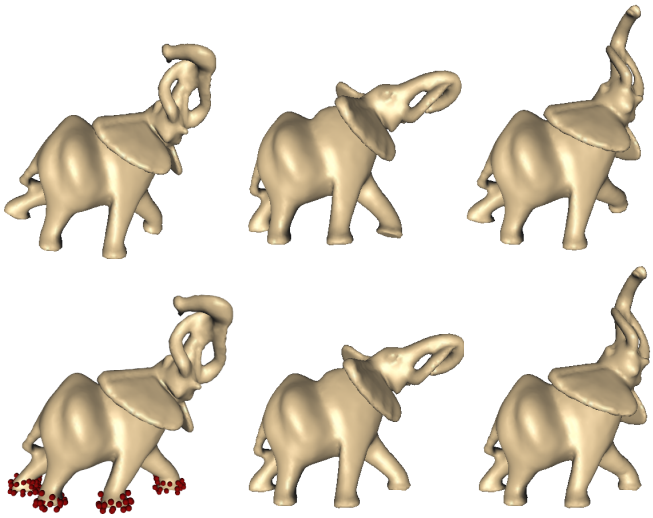

Figure 7: *The feet of the elephant will not stay firmly on the ground if we only use barycenter position constraint in Poisson reconstruction (top row). Position constraints (the red dots) can thus be used to control the placement of the feet.*

tives at t_{i-1}, t_i and t_{i+1} to 0. This simple procedure offers additional control over the results as Figure 8 demonstrates. As shown in Figure 9, 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. 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 [Sumner and Popović 2004; Xu et al. 2005; Kilian et al. 2007], or [Adams et al. 2008] for instance) and *add 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: $\underline{\widehat{X}}=\{\log R(\underline{X})_i, S(\underline{X})_i\}$). Now the only changes in our dynamic interpolation procedure are to substitute $\widehat{X}(t) = (1-\gamma)\underline{\widehat{X}} + \gamma\widehat{W}z(t)$ (γ is used to determine

Figure 8: *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.*

the amplitude of the vibrations) for Eq. (6) and to modify the shape reconstruction to take the given quasi-static path into account by solving:

$$\arg\min_x \|Gx - [(\exp(\gamma \log(R_i) + (1-\gamma)\log R(\underline{X})_i) (\mathrm{Id} + \gamma S_i + (1-\gamma) S(\underline{X})_i)]\|^2$$
$$\text{subject to } \frac{1}{n}\sum_{i=1}^{n} x_i = c.$$

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.

Other Vibration Modes Although we used the vibration modes resulting from modal analysis in our exposition, 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 [Grinspun et al. 2003] as shown in Figure 10. 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 [Sumner and Popović 2004]. 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.

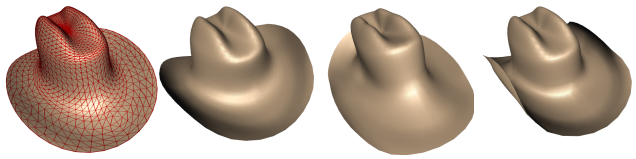

Figure 10: *We can interpolate between triangle meshes as well, using a deformation model governed by thin-shell elastic energy. The rest shape is (a). The key frames in this example are (a) and (d). (b) and (c) are intermediate frames in the animation sequence obtained by our method.*

4 Implementation Details and Performance

4.1 Rotation-Strain Coordinates under 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 salient 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 tet element. When we find an element with its unit rotation axis differing greatly from those of its (traversed) neighbors (the 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.

4.2 Initial value for 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 of the 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.

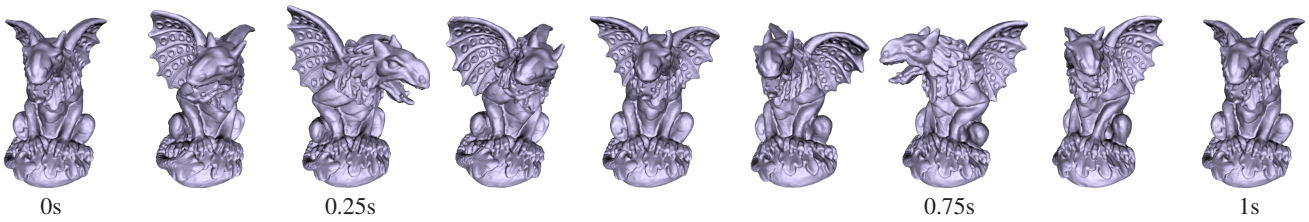

Figure 9: *Our rotation-strain formulation allows us to use Wiggly Splines to interpolate between four key frames (given at time* $0, 0.25, 0.75$ *and* 1 *s.), where both the first and the last pose are the rest state.*

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 necessarily 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 pre-computations 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 (a tetrahedron generation software [Schöberl 1997]) to tessellate the enlarged simplified mesh into the tetrahedron mesh. The triangle meshes used as key frames and outputs can be interpolated from the deformed tetrahedron mesh. A dense tessellation may be required to capture high frequency vibration modes (e.g. the dragon's horn and beard). We can however deal with 17K tet meshes still interactively, which proved sufficient for the design of dynamical effects in this paper.

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.

Limitations of our method include the sacrifice of physical accuracy for speed, and the restriction to a spectral motion subspace. For fast-moving skeleton-driven animation, inertial forces may need to be taken into account in the form of modal forces. It would also be interesting to explore the extension of the motion subspace through modal derivatives [Barbič and James 2005].

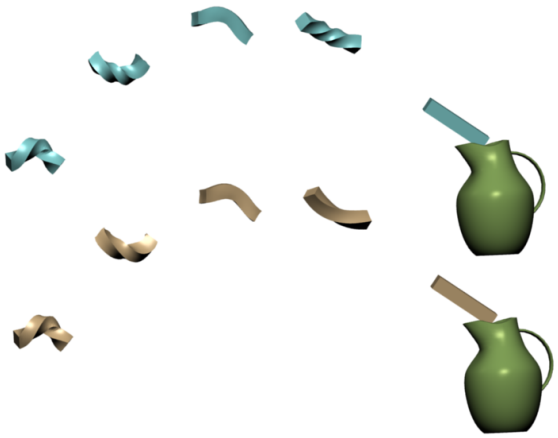

Figure 11: *The motion of a rubber stick (straight at rest) thrown into a jar can be interactively edited to achieve various cinematic effects. In this example, the bottom row uses a larger damping coefficient for the mode leading to a twist-like deformation.*

References

ADAMS, B., OVSJANIKOV, M., WAND, M., SEIDEL, H.-P., AND GUIBAS, L. J. 2008. Meshless modeling of deformable shapes and their motion. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 157–164.

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (Aug.), 982–990.

BARBIČ, J., AND POPOVIĆ, J. 2008. Real-time control of physically based simulations using gentle forces. *ACM Trans. Graph. 27*, 5, 1–10.

BATHE, K. J. 1995. *Finite Element Procedures in Engineering Analysis*. Prentice Hall, Englewood Cliffs, N. J.

BERTRAM, A. 2005. *Elasticity and Plasticity of Large Deformations: An Introduction*. Springer.

CHOI, M. G., AND KO, H.-S. 2005. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics 11*, 1, 91–101.

DAVIS, T. A. 2004. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw. 30*, 2, 196–199.

DE VEUBEKE, B. F. 1976. The dynamics of flexible bodies. *International Journal of Engineering Science 14*, 895–913.

DER, K. G., SUMNER, R. W., AND POPOVIĆ, J. 2006. Inverse kinematics for reduced deformable models. *ACM Trans. Graph. 25*, 3, 1174–1179.

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
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
hat	1607	4771	3164	50	7s	6ms	9ms	25ms	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.*

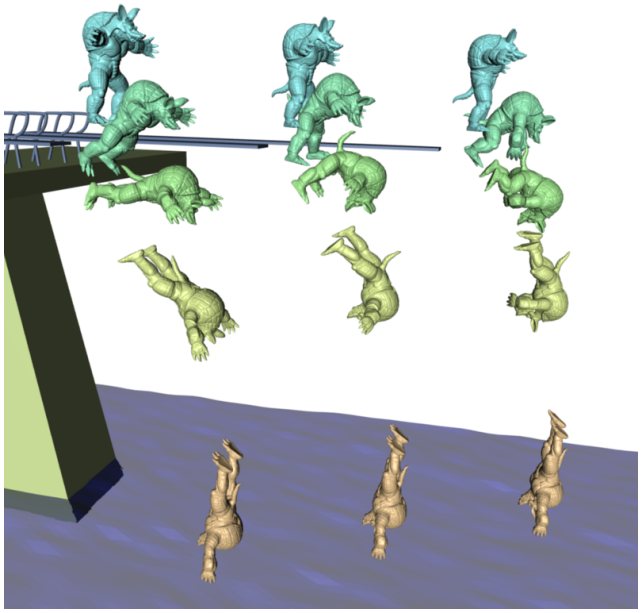

Figure 12: *Synchronized armadillo diving team: adjusting frequency & damping parameters and adding key frames lead to various flight styles along the dive of an armadillo. The left column is the result by smooth, vibration-free interpolation. The middle one shows moderate dynamical effects. With one intermediate keyframe (the 3rd model from the top of the right most column), we add a tuck motion to the dive.*

FENG, W.-W., KIM, B.-U., AND YU, Y. 2008. Real-time data driven deformation using kernel canonical correlation analysis. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, 1–9.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 62–67.

HAUSER, K., SHEN, C., AND O'BRIEN, J. F. 2003. Interactive deformations using modal analysis with constraints. In *Proceedings of Graphics Interface 2003*, 247—256.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3, 1134–1141.

JAMES, D. L., AND PAI, D. K. 2002. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 582–585.

KASS, M., AND ANDERSON, J. 2008. Animating oscillatory motion with overlap: wiggly splines. *ACM Trans. Graph. 27*, 3, 1–8.

KILIAN, M., MITRA, N. J., AND POTTMANN, H. 2007. Geometric modeling in shape space. *ACM Trans. Graphics 26*, 3. Proc. SIGGRAPH.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, 49–54.

NEALEN, A., MUELLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (December), 809–836.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: modal dynamics for graphics and animation. In *ACM SIGGRAPH Proceedings*, 215–222.

POPA, T., ZHOU, Q., BRADLEY, D., KRAEVOY, V., FU, H., SHEFFER, A., AND HEIDRICH, W. 2009. Wrinkling captured garments using space-time data-driven deformation. *Computer Graphics Forum (Proc. Eurographics) 28*, 2.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 11–20.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 209–217.

SCHÖBERL, J. 1997. Netgen - an advancing front 2d/3d-mesh generator based on abstract rules. *Comput. Visual. Sci*, 1:41–52.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3, 399–405.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 159–168.

XU, D., ZHANG, H., WANG, Q., AND BAO, H. 2005. Poisson shape interpolation. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*, 267–274.