Pose-Space Subspace Dynamics

Hongyi Xu Jernej Barbič
University of Southern California

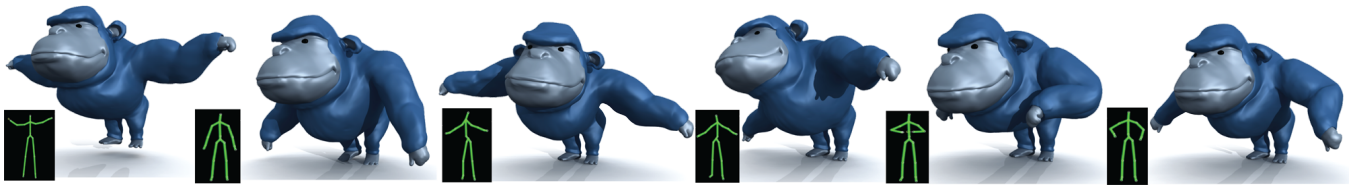

Figure 1: Hard-real-time soft tissue FEM dynamics compatible with standard character rigging. *The input gorilla skeletal (60 DOFs) motion was obtained by retargeting the Kinect-captured motion to the gorilla character in real time. Our method produces physically based FEM dynamics of the gorilla soft tissue. The simulator runs at fast simulation rates (*750 *simulation FPS, 16,297 tetrahedra, 15,744 triangles) and is suitable for applications in games and virtual reality. Speed is achieved using pose-dependent model reduction. We precompute separate reduced models at* 4 *representative gorilla poses. At runtime, simulation is performed in a time-varying basis that is obtained by interpolating the precomputed bases to the current pose. Our method fits into the standard pose-space deformation (PSD) pipeline whereby self-contact and skinning artifacts are resolved, by artists, in each pose and stored as pose-space deformation corrections, and interpolated at runtime. Our subspace is aware of the contact constraints and prohibits dynamics that cause deeper penetrations.*

Abstract

We enrich character animations with secondary soft-tissue Finite Element Method (FEM) dynamics computed under arbitrary rigged or skeletal motion. Our method optionally incorporates pose-space deformation (PSD). It runs at milliseconds per frame for complex characters, and fits directly into standard character animation pipelines. Our simulation method does not require any skin data capture; hence, it can be applied to humans, animals, and arbitrary (real-world or fictional) characters. In standard model reduction of three-dimensional nonlinear solid elastic models, one builds a reduced model around a single pose, typically the rest configuration. We demonstrate how to perform multi-model reduction of Finite Element Method (FEM) nonlinear elasticity, where separate reduced models are precomputed around a representative set of object poses, and then combined at runtime into a single fast dynamic system, using subspace interpolation. While time-varying reduction has been demonstrated before for offline applications, our method is fast and suitable for hard real-time applications in games and virtual reality. Our method supports self-contact, which we achieve by computing linear modes and derivatives under contact constraints.

Keywords: physically based simulation, character rigging, pose-space, model reduction, FEM, secondary motion, real-time

Concepts: •**Computing methodologies** → **Physical simulation;**

SIGGRAPH '16 Technical Paper,, July 24 - 28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925916

1 Introduction

In computer animation practice, characters are typically animated by first animating the skeleton or rig parameters, either by hand or by using a data-driven technique (motion capture). The polygonal mesh of the character is then deformed kinematically, using a rigging or skinning approach. In order to animate the bulging of muscles, correct the artifacts of linear blend skinning, or sculpt arbitrary, pose-dependent modifications to standard skinning, it is common to use the technique of pose-space deformation (PSD). In PSD, one sculpts pose-dependent corrections to rigging/skinning, and then interpolates them to arbitrary poses using radial-basis functions. PSD is, however, a static technique: for a given pose, it always returns the same shape. In this paper, we investigate how to incorporate physically based simulation into rigging/skinning and/or PSD, to automatically produce secondary skin motion. We present a method that has the following properties: (1) it runs at under 1-2 msec per frame for complex models and is as such suitable for real-time applications in games and virtual reality, (2) it uses the Finite Element Method (FEM) to give simulations a non-jiggly, "solid" look, (3) it works with an arbitrary skinning/rigging method, (4) supports character self-collisions, such as in the elbow and shoulder regions, and (5) it fits into standard computer animation pipelines. Our skin dynamics is driven by the inertial forces due to the skeleton or rig-induced motion, or by gravity or other external forces. Our method uses physically based simulation and does not require scanning skin data from real subjects. As such, it is suitable for cartoon characters, animals, fantasy creatures, in addition to humans.

Our technique works by combining the Finite Element Method with pose-dependent model reduction. Model reduction is a technique wherein high-dimensional equations of motions are projected to a suitable, more manageable low-dimensional space. Model reduction has been commonly employed in computer animation to accelerate physically based simulations. In standard model reduction of three-dimensional nonlinear solid elastic models, however, one builds a reduced model around a single pose, typically the rest configuration. Such a basis becomes inaccurate as the character pose deviates from the rest configuration, due to the changing geometric shape under skinning or rigging, and pose-dependent, artist-sculpted changes in character geometry and material properties. Standard model reduction also suffers from non-locality of

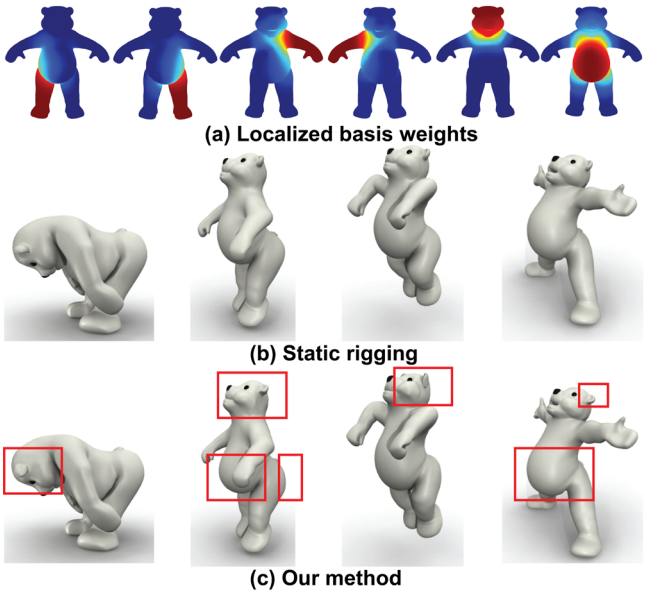

Figure 2: Fast secondary dynamics for keyframed rigged animation. *Given the input keyframed rigged animation created by an artist (*144 *rig DOFs, (b)), our method performs the reduced FEM simulation in real time (*700 *FPS, 12,762 tetrahedra, 6,876 triangles), and produces physically based secondary tissue dynamics, (c). We partition the polar bear tetrahedral mesh into* 6 *overlapping regions, (a), and compute a local basis for each region. Because the regions overlap, their dynamics is coupled automatically and seamlessly, without any constraints or special treatment. Our local bases are interpolated among* 8 *poses and produce rich local and global dynamics on the belly, arms, hips, legs, ears and cheeks.*

deformations, caused by the spatially global modes. Furthermore, self-collisions introduce constraints that greatly change the simulation basis and the dynamic behavior.

Our method addresses these practical challenges, as follows. During preprocess, we build separate reduced models around a representative set of character poses. These poses may incorporate artist-sculpted deformations, as well as pose-dependent material properties to simulate, say, muscle bulging. At runtime, we combine these reduced models into a single fast dynamic system, using subspace interpolation. We drive the motion of the soft tissue using inertial forces arising from skeletal motion or time-varying rig parameters, or external forces. Figure 3 gives an overview of our system. We demonstrate how to align and smoothly interpolate the low-dimensional deformation spaces so that the resulting dynamics is smooth. Time-varying model reduction has been demonstrated before; but with bases that are computed at runtime and not precomputed, which has limited the applications to offline simulation. We present the first practical method to interpolate pose-dependent *pre-computed* simulation spaces, separated by large deformations, into a global fast system. We also demonstrate how to address pose-space self-contact, by computing linear modes under linear contact constraints, to which we propose an efficient preconditioner. Furthermore, we demonstrate how to spatially localize the subspaces, so that the common "ringing" artifacts of spatially global bases are avoided.

Our method is fast and suitable for high-update rate real-time applications in games and virtual reality. We experimentally demonstrate that our model reduction method performs substantially better than coarse full simulations with equal computational budget. Similarly,

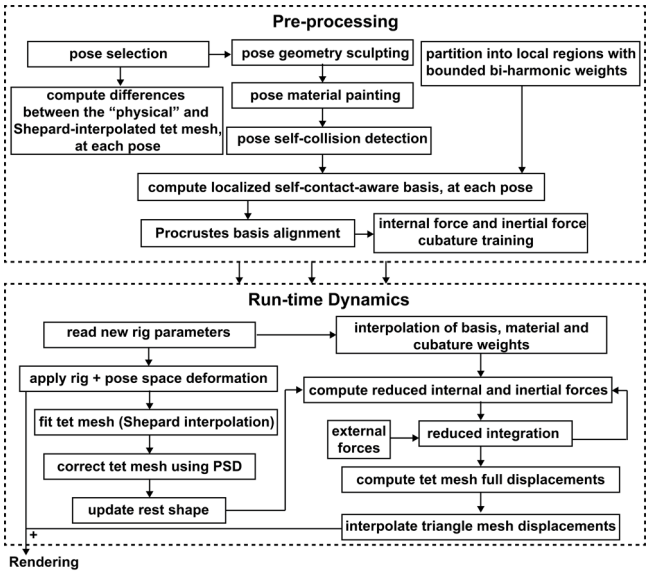

Figure 3: Overview.

we demonstrate that our method outperforms spatially global bases, a single global pose-independent basis, and methods that do not incorporate self-contact into the basis.

2 Related Work

Rigging can generally be defined as a nonlinear mapping between a low-dimensional space of rig parameters and a high-dimensional surface mesh [Hahn et al. 2012]. Common examples include linear blend skinning, and the various nonlinear deformers in popular software animation packages. Enriching rigged motion with dynamics has been widely explored. Several methods drive an unreduced physically based simulation of a solid simulation mesh by constraining it to the underlying skeleton [Capell et al. 2002; Capell et al. 2005; Galoppo et al. 2007; McAdams et al. 2011; Kim and Pollard 2011; Liu et al. 2013]. Such full-space methods do not run at hard real-time rates for complex geometry. In our work, we present a hard real-time simulation approach that trades accuracy for speed using model reduction. Furthermore, we can accommodate general rigging functions instead of just skeleton rigging.

The "**rig-space physics**" methods [Hahn et al. 2012; Hahn et al. 2013] also aim at general rigging functions, and obtain the secondary motion by an optimization in the rig space. Our work differs from rig-space physics as follows. First, our simulation runs entirely in the reduced space and does not require calculating full-space internal forces and their tangent stiffness matrices or projecting them at every simulation step; this difference alone brings us at least a 240× speedup in the gorilla example (Figure 1), compared to the fast rig-space physics method [Hahn et al. 2013]. Second, the dynamics of rig-space physics is limited to the rig space. If the artist did not specifically craft a rig parameter for a certain secondary deformation, that dynamic will not appear in the output. In our work, we generate our bases automatically, and hence our method can automatically generate natural secondary motion for the given geometry, and material properties, without a need for the artist to specifically design for it. At the same time, artists can design for specific dynamic deformations if so required. Our bases automatically incorporate spatially non-homogeneous and pose-varying material properties, whereas the rig-space physics subspace is single-pose and geometric: it does not automatically adapt to non-homogeneous material properties. Third, our method incorpo-

rates a fast self-contact handling method, by baking the per-pose contact state into the per-pose basis.

Subspace methods have been popular in accelerating simulations of deformable solids [James and Pai 2002; Hauser et al. 2003; Barbič and James 2005; An et al. 2008; Hildebrandt et al. 2012]. While fast, it is not immediately obvious how to apply such methods to character animation pipelines that typically use rigging or skinning. Model reduction has proven useful to increase performance for static pose-space deformation without dynamics [Kry et al. 2002], and for dynamic simulation [Galoppo et al. 2009; Hahn et al. 2014; Teng et al. 2015]. A common theme in these papers is to un-transform the simulation data with the skinning transformation to the neutral character pose, and then model-reduce it. However, these prior methods evaluated reduced dynamics with respect to the neutral pose of the character, which is inaccurate for characters undergoing large deformations. Our pose-dependent bases, and the reduced elastic internal forces and stiffness matrices, are computed with respect to the geometry and material properties of each pose, which substantially improves dynamics around poses that contain large deformations. Given several morph targets, Galoppo et al. [2009] constructed a single pose-independent (also called pose-global) basis by performing PCA on the sets of bases computed at the undeformed configuration, but with per-pose material properties and PSD corrections. However, for poses that differ significantly from the undeformed configuration, bases computed at the undeformed pose will produce incorrect dynamics. Different from them, we compute a separate basis at a set of selected poses. Because we never construct a pose-global basis, each of our per-pose bases can have a small dimension, enabling faster simulation rates. At runtime, we interpolate the precomputed bases to the runtime poses. Therefore, our simulator uses a smaller basis, while obtaining more plausible dynamics.

High-quality skin simulations can be achieved using data-driven methods, combined with second-order auto-regressive models [Pons-Moll et al. 2015]. However, such a method requires a real-life subject capture session, and is as such mostly limited to humans. Our simulation method can be used to animate arbitrary creatures, beyond humans. With our method, the animator can also easily tweak the material properties of the tissue, and as such selectively adjust the dynamics at the various parts of the creature. Several simulation methods obtain the basis by performing PCA on full simulation data [Kry et al. 2002; Hahn et al. 2014; Teng et al. 2015]. We create pose-dependent bases automatically, with standard modal analysis techniques, based on geometry and material properties only, and without any training data. To accommodate large deformations, one can enrich the basis using modal derivatives [Barbič and James 2005], or linear transformations of the basis [von Tycowicz et al. 2013]. Our approach is agnostic to the specific enrichment method; we choose linear modes and modal derivatives. For fast modal integration, radial basis interpolation of cubic force polynomials for the St.Venant-Kirchoff material [Galoppo et al. 2009], or pose-space cubature interpolation of contact forces [Teng et al. 2014] have been studied. We demonstrate how to perform pose-space cubature interpolation of general hyperelastic nonlinear materials, with different geometry and materials at each pose. We also demonstrate how to employ cubature to efficiently evaluate the reduced inertial forces.

Our method modifies the simulation basis at runtime. Kim et al. [2009] proposed an online model reduction method which alternates between full and reduced simulation. Our approach always uses reduced simulation, without the need for any runtime full simulation, which makes it possible to run the simulation consistently at hard real-time rates. Temporally adaptive bases can also be constructed by selecting a few basis vectors from a large precomputed database [Hahn et al. 2014]. Their approach, however, requires the evaluation and projection of full-space internal forces and stiffness matrices, which we can avoid. Furthermore, in order to avoid popping, their method requires re-projecting the deformation at each frame to a new basis. Our method creates the subspace by interpolating aligned basis vectors, which requires no re-projection and therefore avoid re-projection errors and loss of energy.

In the engineering community, interpolation between parameterized reduced-order models has been explored for aeroelasticity, thermal design and probabilistic analysis [Amsallem and Farhat 2008; Degroote et al. 2010; Amsallem and Farhat 2011]. In these methods, the varying parameter is typically a flow constant, such as the Mach number, whereas the structural deformations are small and often linearized. In our work, we interpolate bases for geometric shapes that have undergone large deformations. Different from these methods outside of computer graphics, we demonstrate how to construct a system that runs at hard real-time rates for complex three-dimensional geometry undergoing large deformations.

Teng et al. [2014] computed self-contact forces in the subspace with pose-space cubature, by exploiting contact coherence for articulated bodies. Similarly, we detect self-contact at example poses, but then bake the contact information into the construction of our pose-dependent bases. This avoids the need for run-time self-collision detection and resolution, trading accuracy for performance for hard real-time applications. We also demonstrate how to support localized deformations using spatially localized basis functions. Local subspace deformations can be accommodated with the use of analytic Boussinesq solutions [Harmon and Zorin 2013], bi-harmonic weights [Jacobson et al. 2011], discrete Laplacian [Wang et al. 2015], sparse matrix decomposition of animation sequences [Neumann et al. 2013], or domain decomposition [Barbič and Zhao 2011; Kim and James 2012; Yang et al. 2013]. Huang et al. [2012] trained spatially-local skinning deformation mappings in the local pose space. In contrast to prior work, our localized deformations are designed for physical simulation involving large deformations induced by a character rig, and are free of seaming artifacts. We construct our localized bases by letting the user specify a few points to denote individual regions, upon which a scalar "region function" is computed for each region, using bounded bi-harmonic weights [Jacobson et al. 2011]. Different from [Jacobson et al. 2011] who used bi-harmonic functions directly as the deformation subspace for static shape editing, we only use them to define overlapping regions for model reduction, and simulate real-time dynamics.

3 Background: Pose-Space Deformation

In our work, we add physically based secondary motion effects, in real-time, to triangle mesh animations obtained using any character rigging process. In order to do so, we will employ a simulation tetrahedral mesh. We denote all quantities y referring to the triangle and tetrahedral meshes as $\bar{y}$ and y, respectively. The input to our method is an undeformed (also called neutral) triangle mesh $\bar{\Gamma} \in \mathbb{R}^{3\bar{n}}$ with $\bar{n}$ vertices with positions $\bar{X} \in \mathbb{R}^{3\bar{n}}$, alongside with an arbitrary rigging function $\bar{\Phi}(p,\bar{X})$. Here, $p \in \mathbb{R}^s$ is the rig parameter which defines the specific pose. The rig function $\bar{\Phi}(p,\bar{X}) \in \mathbb{R}^{3\bar{n}}$ gives the rig-deformed vertex positions; these positions are statically determined by p, devoid of any dynamics. Note that the pose p may correspond to the joint angles of the character, but it can also be more general; e.g., sliders in a rigging deformer, or even any more abstract space. Our general formulation for $\bar{\Phi}$ incorporates elaborate character "production" rigs, nonlinear deformers, skeleton-based methods (linear blend skinning, dual quaternions), and blend-shape animations. In our system, the rigging function $\bar{\Phi}$ is treated as a black-box, and we do not require its explicit formula; we only need the ability to evaluate $\bar{\Phi}(p,\bar{X})$ for an arbitrary p and

$\bar{X}$. This allows us to use our method with, say, standard animation packages such as Autodesk Maya.

Pose-space Deformation is a method that combines skeleton subspace deformation (SSD) [Magnenat-Thalmann et al. 1988] with artist-corrected pose shapes [Lewis et al. 2000]. Given a set of triangle mesh poses $\bar{S}_i$ corresponding to poses p_i, for $i = 1,\ldots,m$, typically obtained directly by skinning or rigging, the artist provides corrections $\bar{\delta}_i \in \mathbb{R}^{3\bar{n}}$ that, for example, undo a candy wrapper effect or improve volume preservation. We can then incorporate these correction into the rig, by redefining it

$$\bar{\Phi}(p,\bar{X}) \rightarrow \bar{\Phi}(p,\bar{X}) + \bar{\delta}(p). \tag{1}$$

The nonlinear deformation correction $\bar{\delta}(p)$ can be obtained via scattered-data interpolation using Radial Basis Functions (RBF) in the pose-space as

$$\bar{\delta}(p) = \sum_{i=1}^{m} w_i(p)\bar{\delta}_i, \quad \text{for} \quad w_i(p) = \sum_{j=1}^{m} \hat{w}_{ij}\phi(\|p-p_j\|), \tag{2}$$

where $w_i(p) \in \mathbb{R}, i = 1,\ldots,m$, are the normalized interpolation weights, $\hat{w}_{ij} \in \mathbb{R}$ are the RBF-trained weights so that $w_i(p_j)$ is 1 when $i = j$ and 0 otherwise, and ϕ is the RBF kernel function. In our system, we employ the globally supported commonly used biharmonic RBF kernel $\phi(r) = r$, considering the sparsity of the example poses [Carr et al. 2001]. Note that in principle, a distinction in notation should be made between the original rig function, and the rig function that incorporates the PSD corrections. In our paper, we will hereforth simply use $\bar{\Phi}$ to denote the PSD-corrected rig, because we never need to reference the original rig again. Our method works equally well even if there is no PSD correction. Because $\bar{X}$ is constant, we will often drop $\bar{X}$ and simply write $\bar{\Phi}(p)$.

4 Pose-Space Dynamics

How can one define meaningful physically-based dynamics for the rigging function $\bar{\Phi}$? We assume that the trajectory $p = p(t)$ is given externally, as our input. This is compatible with the usual computer animation pipelines, as $p(t)$ can be obtained in any standard way: motion capture, procedural animation, inverse kinematics, etc. Therefore, in departure to prior work on rig-space physics, we chose *not* to modify p or evolve p according to an ODE. Instead, we treat each configuration p as if it was a new rest configuration of the object. The justification for this decision is that in static (non-dynamic) PSD, a pose p *uniquely* defines the shape: if p is kept constant, the shape in static PSD never changes and can as such be seen as the rest shape corresponding to p. A similar view has been proposed by Liu et al. [2013], who used it for unreduced FEM simulation and control, limited to linear blend skinning and the co-rotational material. As $p = p(t)$ evolves over time, the object undergoes a trajectory of rest configurations. For each p, we then simulate dynamics on top of this time-varying rest pose. Note that our dynamic PSD supports, using pose-space interpolation, effects such as materials stiffening in specific poses due to a large strain, or due to activation (muscles). We can simulate any physical force, such as gravity, collision forces or user forces. Additionally, the dynamics are driven by the inertial (also called "system", "fictitious", or "d'Alembert") forces, arising due to the changing rig parameter p. Inertial forces are responsible for most of our dynamics, such as the tissue overshooting when the character stops, or deformations under Coriolis forces due to bone motion. The output of our method is an animation of $\bar{\Gamma}$ that is driven by $p = p(t)$, but is enriched with physically-based secondary motion.

4.1 Pose-Space Tetrahedral Mesh

Our secondary motion originates from a pose-aware model-reduced FEM simulation of the tetrahedral mesh $\Gamma \in \mathbb{R}^{3n}$. During preprocess, we compute Γ by meshing the space enclosed by $\bar{\Gamma}$, in the neutral configuration. Our method accommodates arbitrary, non-manifold triangular geometry $\bar{\Gamma}$, using signed distance field meshing [Xu and Barbič 2014]. The input rig function $\bar{\Phi}$ is defined on the triangle mesh $\bar{\Gamma}$, whereas we run physically-based simulations on the tetrahedral mesh Γ. Therefore, we need to define the rig function Φ on the tetrahedral mesh. For each tet mesh vertex i, we perform this by interpolating the displacements of the nearest k triangle mesh vertices (we use $k = 4$ in our method), using Shepard "inverse-distance" weights [Barnhill et al. 1983]. The interpolation weights are determined during the pre-process, in the neutral configuration. Such a method may produce a non-smooth deformation of Γ, when the vertex distribution of $\bar{\Gamma}$ is non-uniform. Therefore, in spirit of PSD, for each pose p_i, we first compute a "physical" well-fitted tet mesh, using optimization [Barbič et al. 2009]

$$\Phi(p_i) = \arg\min_{\hat{x}} \Big(\|A\hat{x} - \bar{\Phi}(p_i)\|_M^2 + \gamma \mathcal{E}(\hat{x}) \Big). \tag{3}$$

Here, A is the sparse barycentric interpolation matrix between Γ and $\bar{\Gamma}$, determined in the neutral pose, $\mathcal{E}(\hat{x})$ is the elastic strain energy for tet mesh vertex positions $\hat{x}$ with respect to the neutral configuration, and $\gamma > 0$ is a regularization constant. We store the difference between the Shepard-interpolated mesh and $\Phi(p_i)$ as corrections $\delta = \{\delta_1, \delta_2, \ldots, \delta_m\}$, at each pose. At run time, we correct the Shepard-interpolated tet mesh with a PSD-interpolated correction $\delta(p) = w(p)\delta \in \mathbb{R}^{3n}$. We note that the "rig-space physics" method [Hahn et al. 2013] addresses a similar technical issue by training weights relating displacements of surface simulation vertices to the interior vertices, using static simulation. While such a method could be used to further improve the fit, the PSD scheme above is very fast, and avoids the need for a weight training process.

4.2 Equations of Motion

Our dynamics is layered on top of the triangle mesh driven by the rig parameter p. At any frame, we treat the current rigged mesh $\Phi(p)$ as the rest shape. The dynamic deformation $u \in \mathbb{R}^{3n}$ then gives the displacements away from this rest shape, in world coordinates. We write the equations of motion for the tetrahedral mesh as

$$M(p)\ddot{u} + D\dot{u} + f_{\text{int}}\big(\Phi(p),u\big) = f_{\text{ext}} + f_{\text{inert}}, \tag{4}$$

where $M(p) \in \mathbb{R}^{3n\times 3n}$ is the mass matrix, $D \in \mathbb{R}^{3n\times 3n}$ is the damping matrix, $f_{\text{int}}(x,u) \in \mathbb{R}^{3n}$ is the internal force under the rest shape x and displacements u away from x, $f_{\text{ext}} \in \mathbb{R}^{3n}$ are the external forces and $f_{\text{inert}} \in \mathbb{R}^{3n}$ are the inertial forces caused by the motion $p = p(t)$. We use the Rayleigh damping model $D = \alpha M + \beta K(\Phi(p),u)$, where $K(\Phi(p),u) \in \mathbb{R}^{3n}$ is the tangent stiffness matrix with respect to the current rest shape. Vector u gives the displacements relative to $\Phi(p)$, expressed in the world-coordinate system. We add the inertial force f_{inert} to model the effect of the changing rig parameter p onto the deformation u. Consider a sequence of tet mesh rest poses $x(p)$, under a time-varying p. Then, any location in the material, at any time t, undergoes a world-coordinate acceleration $\ddot{x}(p)$ with respect to the world-coordinate system, simply because of the rig-induced motion. Therefore, from the point of view of a non-inertial coordinate system attached to a small volume dV in the material, the material experiences an inertial force $-\rho\ddot{x}(p)dV$, where ρ is mass density. The inertial force on the tet mesh vertices equals $f_{\text{inert}} = -Ma(p) = -Md^2x(p)/dt^2$. We show how to efficiently evaluate f_{inert} in Section 5.4. We compute the final deformed position of the triangle mesh $\bar{\Gamma}$ as $\bar{\Phi}(p) + Au$, as

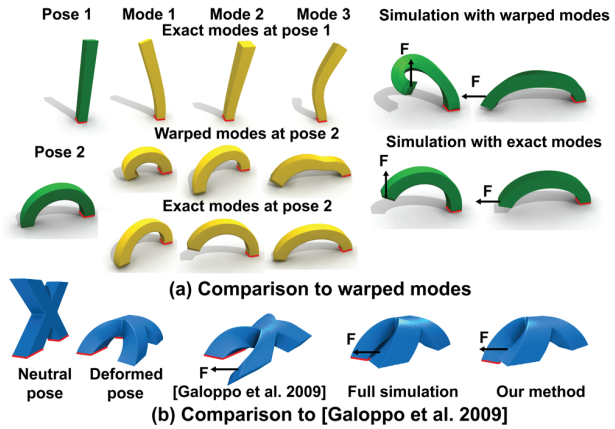

Figure 4: The modes change significantly under large deformations. *(a) We compare the modes for a cantilever beam, and cantilever beam twisted into a "U-shape". Reduced models computed in pose 1 are vastly inexact when simply geometrically transferred to pose 2 (middle). If they are used for simulation, unnatural deformations appear (right). For this experiment, we statically loaded the beam with two representative force loads. The warped modes results are obviously wrong. (b) Transforming the deformations back to the neutral shape, and computing dynamics using the basis computed in the neutral pose [Galoppo et al. 2009] also produces incorrect simulation results.*

opposed to $A(\Phi(p)+u)$. Such a choice uses the quality rig shape $\bar{\Phi}(p)$, and makes our system more forgivable to errors in the online-fitted tet mesh $\Phi(p)$, even when Γ does not perfectly enclose $\bar{\Gamma}$.

5 Reduced Pose-Space Dynamics

Equation 4 is too slow for interactive systems. In order to accelerate the computation, we employ model reduction, making the dynamic simulation independent of the mesh size. Such a model reduction, however, must incorporate the fact that we are dealing with a time-varying rest shape. Reduced models computed under one rest shape are vastly incorrect under a new rest shape, if simply geometrically transferred, say, using the local deformation gradients or the rigging function (Figure 4). The errors occur because both the basis and the reduced internal forces change substantially under large rest shape changes. They are not simply rotated, or transformed with a deformation gradient, or any similar geometric transformation. Our problem therefore becomes one of generating a sufficient number of reduced models $\Phi(p_i)$, and properly combining them at runtime, based on the current p. Assuming that a good basis $U(p) \in \mathbb{R}^{3n\times r}(r \ll 3n)$ for a rig shape $\Phi(p)$ is known, then, for a fixed p, the ODE in Equation 4 can be projected to

$$\tilde{M}\ddot{q}+\tilde{D}\dot{q}+\tilde{f}_{\text{int}}\Big(\Phi(p),U(p)q\Big)=\tilde{f}_{\text{ext}}+\tilde{f}_{\text{inert}} \tag{5}$$

$$\tilde{D}=\alpha\tilde{M}+\beta\tilde{K}, \qquad \tilde{M}=U^T(p)MU(p), \tag{6}$$

$$\tilde{K}=U^T(p)K\Big(\Phi(p),U(p)q\Big)U(p), \tag{7}$$

$$\tilde{f}_{\text{int}}\Big(\Phi(p),U(p)q\Big)=U^T(p)f_{\text{int}}\Big(\Phi(p),U(p)q\Big), \tag{8}$$

$$\tilde{f}_{\text{ext}}=U^T(p)f_{\text{ext}}, \qquad \tilde{f}_{\text{inert}}=U^T(p)f_{\text{inert}}, \tag{9}$$

where q are the reduced coordinates. In order to form a new basis $U(p)$ at runtime, one could solve the generalized eigenproblem $Ku=\lambda Mu$, and then enrich the linear modes with modal derivatives [Barbič and James 2005]. Solving such a generalized eigenproblem is quite expensive, however, and not suitable for runtime

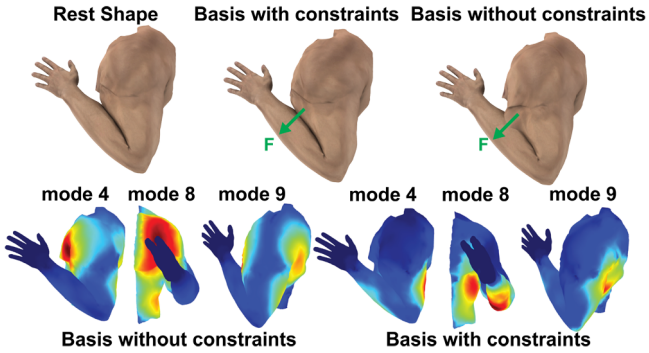

Figure 5: Reduced models under contact. *The human arm pose (top left) has* 197 *contact points between the upper arm, forearm and the body. Our basis computed with linear constraints effectively prevents self-penetration during the motion, while basis without constraints cannot (top right). The second rows compares the modes computed without and with contact constraints (blue: low deformation magnitude; red: high deformation magnitude). The reduced modes are significantly different when contact is involved.*

computation. Moreover, without preprocessing the reduced internal forces, such as by using cubature [An et al. 2008] or StVK cubic polynomials [Barbič and James 2005], computing the reduced internal forces, mass matrix and the reduced tangent stiffness matrix for implicit integration are orders of magnitude too slow for real-time simulation of complex meshes.

Another challenge that we must address is that, generally, when a basis is changed, popping may occur. To remove the popping artifacts when transforming to a new basis, careful design of the basis [Hahn et al. 2014] is generally needed. Re-projecting the full-space state variables $(u,\dot{u},\ddot{u})$ to obtain new reduced state vectors $(q,\dot{q},\ddot{q})$ is also necessary in previous methods [Kim and James 2009; Harmon and Zorin 2013; Hahn et al. 2014], an operation which introduces a certain amount of energy loss (artificial damping). One approach to avoid these issues is to construct a pose-independent basis using Principal Component Analysis. This has been done either by using full simulation data [Teng et al. 2014], or by combining bases computed at shapes obtained by perturbing the neutral configuration with the morph targets [Galoppo et al. 2009]. However, because these methods ultimately use one big, pose-independent global basis, the number of required modes needed to accommodate a wide range of rig shapes will typically be quite large. A pose-global basis is especially problematic when material properties, and, crucially, contact configurations, change with p (Figures 5, 14). In our work, we do not form a pose-independent basis. Instead, we construct a local basis, and a reduced model, at each pose p_i, and then *interpolate* these bases and reduced models at runtime.

Similar to PSD sculpting, we leave the pose selection to the artist. We build a reduced model at each PSD-corrected pose (plus the neutral pose) p_i, for $i=1,\ldots,m$. We choose these poses because PSD corrections are usually needed at extreme poses where geometry undergoes significant changes, and hence these are natural choices for where a modified reduced dynamic model is also needed. If additional reduced models are needed at poses where a PSD correction was not provided, such as at poses with significant material or contact configuration changes, we simply increase m to include such poses, with an interpolated PSD correction. For each mesh pose p_i, $i=1,\ldots,m$, we first fit the corresponding tetrahedral mesh pose $\Phi(p_i)$, using Equation 3. We then construct a basis $U_i \in \mathbb{R}^{3n\times r}$, by first solving the generalized eigenproblem (for a

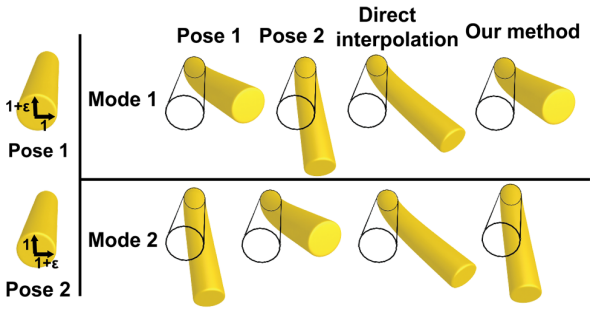

Figure 6: Basis alignment. *The basis vectors may permute, or change completely, due to shape changes, varying contact configurations and material property changes. In this case, the two poses are almost identical with only a small perturbation ε on the width and height of the beam. The first two modes permute. When ε is close to 0 and we do interpolation with weights $(0.5, 0.5)$, direct linear interpolation of basis columns, followed by normalization, results in two identical modes (degenerated subspace). Our alignment procedure (right) produces a good interpolated basis.*

small number of eigenvalues $\hat{r}$)

$$K\big(\Phi(p_i)\big)U_i^{\text{lin}} = MU_i^{\text{lin}}\Lambda, \tag{10}$$

where $\Lambda \in \mathbb{R}^{\hat{r}\times\hat{r}}$ is a diagonal matrix consisting of the first $\hat{r}$ eigenvalues. To accommodate large deformations around each pose, we augment U_i^{lin} with modal derivatives [Barbič and James 2005]. We use the same basis size r for all of our poses. We also mass-orthonormalize our bases, $U_i^T MU_i = I$.

It is not immediately obvious how to obtain a basis $U(p)$ for some arbitrary pose p, based on known bases U_i, how to then compute the reduced internal forces, stiffness matrices and inertial forces, and do so rapidly even for complex models, so that the method is useful for real-time applications in games and virtual reality. To the best of our knowledge, we are the first work in interactive simulation of nonlinear elastic solids to undertake such a challenge. We now describe our solution: pose-space basis alignment and interpolation procedure, which addresses these challenges while still maintaining hard real-time performance.

5.1 Basis Alignment

At runtime, for a given new pose parameter p, we construct the reduced basis $U(p)$ by interpolating the pose bases. How to interpolate two or more bases? A naive approach is to linearly interpolate each column of the basis matrix, optionally followed by normalization. These approaches fail because columns with the same index i do not necessary match each other in any meaningful way. This is especially pronounced when the deformation is substantial, or the contact configuration or material properties change. In such cases, the basis content changes fundamentally, and there is no obvious correspondence between the basis vectors. The naive interpolation approach produces visible artifacts (Figure 6).

Given two reduced bases U_i and U_j defined on a common underlying mesh, we instead ask, what is the natural, minimal transformation of the *linear subspace* spanned by the columns of U_i into the *linear subspace* spanned by the columns of U_j? Such a transformation can be discovered by solving for congruence rotations $Q_i, Q_j \in \mathbb{R}^{r\times r}$, such that U_iQ_i can be interpolated into U_jQ_j simply by linearly interpolating the columns. Because of symmetry, we can fix one of the basis transformations to be identity ($Q_i = I_r$), and then Q_j can be obtained by solving an orthogonal Procrustes

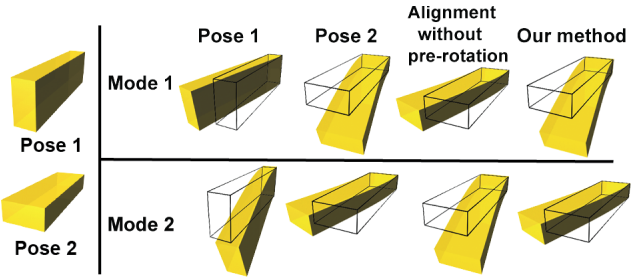

Figure 7: Mesh alignment before Procrustes alignment. *Procrustes-only alignment of bases between two meshes with equal topology, but distinct vertex positions, can easily produce incorrect results, as modes may be matched to incorrect modes. In this case, pose 1 and 2 are rotated by 90 degrees with respect to each other, but are otherwise identical. Hence, the modes for pose 1 and 2 are the same, but rotated by 90 degrees. Without mesh pre-alignment, the Procrustes basis alignment will yield a mode permutation. In our method, we first un-rotate the second basis with the local geometric rotation between the two meshes, then perform Procrustes alignment, and rotate back, which greatly improves the alignment.*

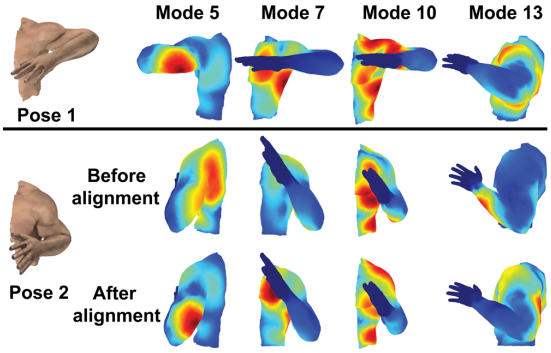

Figure 8: Basis alignment on a complex mesh with self-contact: *Given bases matrices $U_i \in \mathbb{R}^{3n\times r}$ computed at poses $i = 1, 2$, and incorporating self-contact constraints, our Procrustes alignment finds a basis U_2Q (for some $Q \in \mathbb{R}^{r\times r}$) spanning the same subspace as U_2 at pose 2, which optimally aligns with U_1. After alignment, basis matrix columns can be linearly interpolated.*

problem [Gower and Dijksterhuis 2004], minimizing

$$Q_j = \underset{Q\in\mathbb{R}^{r\times r}}{\arg\min} \|U_jQ - U_i\|_M, \qquad \text{subject to } Q^TQ = I_r. \tag{11}$$

The optimal transformation Q_j can be determined analytically as

$$U_j^T MU_i = Y\Sigma Z^T \text{ (SVD)}, \qquad Q_j = YZ^T, \tag{12}$$

where $Y = [y_1, y_2, \ldots, y_r] \in \mathbb{R}^{r\times r}$, $Z = [z_1, z_2, \ldots, z_r] \in \mathbb{R}^{r\times r}$, and $\Sigma = \text{diag}[\sigma_1, \sigma_2, \ldots, \sigma_r] \in \mathbb{R}^{r\times r}$. Figure 8 demonstrates such an alignment result. Note that the subspace angles $\theta_k, k = 1, \ldots, r$, between the two basis vectors U_jy_k and U_iz_k can be computed as $\theta_k = \arccos(\sigma_k)$. A subspace angle $\theta_k = 0$ denotes perfect consistency between the associated vectors. On the other hand, a large subspace angle indicates that the two subspaces are dissimilar. Such a metric can be used to insert an intermediate pose as needed.

In our work, the two bases $U_i = U(p_i)$ and $U_j = U(p_j)$ correspond to poses p_i and p_j, respectively. They are defined on meshes $\Phi(p_i)$ and $\Phi(p_j)$ that have the same connectivity, but different vertex positions. Therefore, the approach from the previous paragraph does not work if there are large rotations between the two meshes. For example, suppose that pose j is obtained from pose i by a global rotation. In such a case, the individual basis vectors at each vertex will

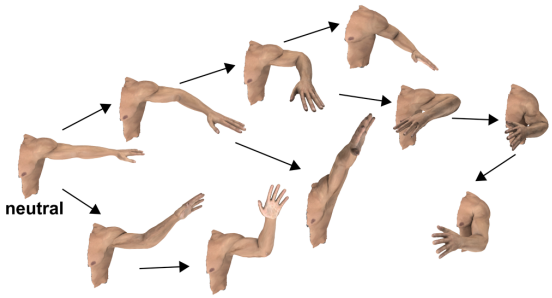

Figure 9: *The alignment graph $\mathcal{G}$ for the human arm example.*

simply be rotated by this global rotation, which will cause the Procrustes procedure to incorrectly align them (Figure 7). We address this issue by aligning the two geometric shapes using local vertex rotations. Before the Procrustes alignment, we first per-vertex-rotate basis U_j, to transform it onto pose i, so that the Procrustes alignment then proceeds on rotation-aligned shapes. To transform U_j to pose i, we compute per-vertex rotations $\mathcal{R}_{ij}^k \in \mathbb{R}^{3\times3}$ from x_i to x_j, for vertices $k = 1,\ldots,n$,

$$\mathcal{R}_{ij}^k = \text{quaternionToRotation}\Big(\frac{\sum_{e\in N_k} q_{ij}^e V^e}{\sum_{e\in N_k} V^e}\Big),\ F_{ij}^e = Q_{ij}^e R_{ij}^e, \quad (13)$$

where N_k is the 1-ring of neighboring elements to vertex k, V^e is the element volume in pose i, and $F_{ij}^e \in \mathbb{R}^{3\times3}$ is the element deformation gradient between poses i and j. We perform polar decomposition to obtain per-element rotations $R_{ij}^e \in \mathbb{R}^{3\times3}$ and convert them to quaternions q_{ij}^e. Then, we rotate vertex k in each basis vector of U_j by $\left(\mathcal{R}_{ij}^k\right)^{-1}$, and perform Procrustes alignment. After the alignment, we then rotate the aligned matrix back to pose j, using $\mathcal{R}_{ij}^k$.

So far, we have discussed alignment between two poses. We now consider the basis alignment among multiple poses. Conceptually, we can form a directed graph $\mathcal{G}$ whose nodes are poses, and two nodes i and j are joined by a directed edge if we perform an alignment between U_i and U_j. Because one rotation $Q_i = I$ in each pair is fixed, the graph must be connected and free of cycles, with each node receiving at most a single incoming connection, but potentially having multiple outgoing connections, i.e., a tree. Given a set of poses, $\mathcal{G}$ can take many forms. Any node can in principle serve as the root, but it is most natural for the neutral pose to be the root. Our alignment strategy proceeds greedily from the neutral pose (root), maintaining a list of nodes that have already been connected in the tree. At any step, it selects the node pair (i, j) that minimizes the distance $||p_i - p_j||^2$, where i and j are poses from the connected and un-connected part of $\mathcal{G}$, respectively. It then aligns j to i and adds j to the connected set. These steps are repeated until the entire $\mathcal{G}$ is connected (Figure 9).

5.2 Basis Interpolation

With all the bases aligned, we can now do runtime basis interpolation to new poses p (Figure 10). In order to preserve the mass orthogonality of the basis, one may choose to interpolate basis on the tangent space of the Grassman manifold [Amsallem and Farhat 2008]. However, doing so requires a logarithmic mapping of the bases onto the Grassman manifold tangent space, and an exponential map of the interpolated basis back to manifold, which are expensive to evaluate at runtime. Mass-orthogonality of the basis is a useful, but not a strictly necessary property for our method. As long as the columns of U are linearly independent, the reduced stiffness matrix $\tilde{K}$ remains invertible. In addition, $\tilde{K}$ is regularized by the

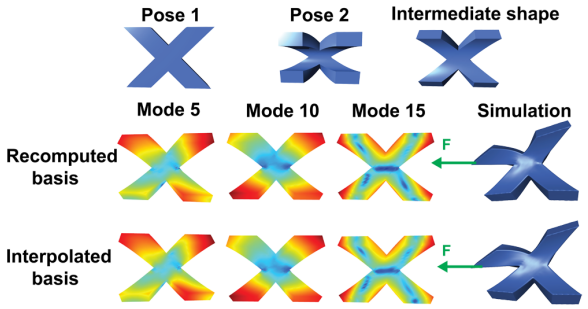

Figure 10: Basis interpolation. *Our basis interpolation between pre-aligned subspaces provides a very good approximation to the subspace recomputed using the exact interpolated geometry. Similarly, our interpolated cubature weights also produce small approximation error. The error in approximating exact forces using cubature for poses* 1 *and* 2 *is* 1.1% *and* 1.9%*, respectively. In the intermediate shape, our interpolated cubature error computed with the interpolated bases is* 1.5% *while recomputed cubature error is* 1.3%.

presence of $\tilde{M}$ in the reduced system. Hence, the reduced simulation can proceed without issues even if the columns of U are not orthogonal. Because we have pre-aligned our bases, we have found that it is in practice sufficient to only enforce a unit mass-norm of each interpolated basis vector. We start by linearly interpolating the bases with PSD scatter-data interpolation weights (Equation 2),

$$\hat{U}(p) = \sum_{i=1}^{m} w_i(p) U_i, \quad (14)$$

where $\hat{U}(p)$ is the interpolated basis before mass-normalization. The mass-norm υ_k of the k-th basis vector can be computed as

$$\upsilon_k^2(p) = \|\sum_{i=1}^{m} w_i(p) U_i^k\|_M^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} w_i(p) w_j(p) \left(U_i^k\right)^T M U_j^k. \quad (15)$$

Then, we form a diagonal matrix $\Upsilon(p) \in \mathbb{R}^{r\times r}$ consisting of $1.0/\upsilon_k(p)$. The normalized basis and reduced mass matrix are

$$U = \hat{U}\Upsilon,\ \tilde{M} = \Upsilon^T \hat{U}^T M \hat{U} \Upsilon = \Upsilon^T \Big(\sum_{i=1}^{m}\sum_{j=1}^{m} w_i w_j U_i^T M U_j\Big)\Upsilon, \quad (16)$$

where we have dropped the dependency of $U, \hat{U}, \Upsilon, w_i, w_j$ on p, for brevity. The matrices $U_i^T M U_j \in \mathbb{R}^{r\times r}, i, j = 1,\ldots,m$, can be precomputed. At runtime, we first evaluate the double sum in Equation 16. By Equation 15, the diagonal of the resulting matrix consists of υ_k^2. We can then form Υ, and evaluate $\tilde{M}$ and $\tilde{f}_{\text{ext}}$. We evaluate quantities $\tilde{K}, \tilde{f}_{\text{int}}, \tilde{f}_{\text{inert}}$ using cubature (Section 5.3). Note that the diagonal entries of $\tilde{M}$ are always 1 in our method. Our construction ensures that the reduced mass matrix and the interpolated basis are always consistent with each other, $\tilde{M}(p) = U(p)^T M U(p)$. The full-space mass matrix M is less sensitive to pose changes than the stiffness matrix. We assume that the mass distribution around every vertex roughly stays unmodified at the different poses, and use a constant mass matrix for simplicity. Pose-dependent $M(p)$ could be accommodated by evaluating Equation 16 using cubature.

In [Kim and James 2009; Harmon and Zorin 2013; Hahn et al. 2014], full-space quantities $(u, \dot{u}, \ddot{u})$ must be re-projected into the new basis each time the basis changes. This is not necessary in our method, and both simplifies the computation and avoids the artificial damping introduced by such a re-projection. We can avoid the re-projection because our displacement vector u is always defined

with respect to the current pose p. Similarly to how the rest shapes are morphed via the rig $\bar{\Phi}(p)$, our basis interpolation can be seen as morphing displacements (away from the rigged shape), as p varies. The basis $U(p)$ defines the current local coordinates at each vertex that interpret the meaning of q, and updating the pose p is equivalent to transforming this "local coordinate system". Furthermore, the bases are pre-aligned and thus it is not necessary to remap the reduced quantities. We mass-normalize the basis and thus magnitudes of the displacements vary smoothly, free of popping artifacts.

5.3 Pose-Space Cubature for Elasticity

We now describe how we quickly evaluate the reduced internal forces $\tilde{f}_{\text{int}}(\Phi(p),U(p)q)$ and reduced tangent stiffness matrix $\tilde{K}(\Phi(p),U(p)q)$, for general nonlinear material, under the interpolated basis $U(p)$, for arbitrary p and q. For the linear StVK material, Galoppo [2009] proposed computing these quantities by the interpolation of cubic polynomial coefficients associated with each pose. We introduce *pose-space cubature* to simulate arbitrary, nonlinear materials, similar in spirit to [An et al. 2008; Teng et al. 2014], but for multi-pose reduced elasticity.

For each pose p_i, we determine a cubature element set $\mathcal{C}_i$, and the corresponding cubature weights $v_i \in \mathbb{R}^{c_i}$, where $c_i = |\mathcal{C}_i|$. At runtime, we then linearly interpolate the cubature weights to each pose p, using PSD weights. In each pose p_i, we generate $\mathcal{T}$ random cubature samples, $q^{(i,k)},\ k = 1,\ldots,\mathcal{T}$. The samples are obtained by randomly sampling a Gaussian distribution for each mode, with standard deviations proportional to the inverse of the modes stiffness [An et al. 2008]. Independent cubature training at each pose would in general result in different per-pose cubature tets. Therefore, during runtime interpolation, the cardinality could grow as large as $\sum_{i=1}^{m} c_i$. To avoid this problem, we use the same set of cubature elements at every pose, $\mathcal{C}_i = \mathcal{C}$. We select this global cubature set $\mathcal{C}$ based on the combined $m\mathcal{T}$ samples at all poses, using Hard Thresholding Pursuit (HTP) [von Tycowicz et al. 2013]. The weights are determined using Non-Negative Least Squares (NNLS) [Chen and Plemmons 2006],

$$\begin{bmatrix} g_1^1 & g_1^2 & \cdots & g_1^c \\ g_2^1 & g_2^2 & \cdots & g_2^c \\ \cdots & & & \\ g_m^1 & g_m^2 & \cdots & g_m^c \end{bmatrix} v = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_m \end{bmatrix}, \quad v \geq 0, \tag{17}$$

where the k-th components of vectors $g_i^e \in \mathbb{R}^{\mathcal{T}}$ and $b_i \in \mathbb{R}^{\mathcal{T}}$, for $k = 1,\ldots,\mathcal{T}$, equal

$$g_{i,k}^e = \frac{U_i^{eT} f_{\text{int}}^e(\Phi(p_i), U_i^e q^{(i,k)})}{\|\tilde{f}_{\text{int}}(\Phi(p_i), U_i q^{(i,k)})\|}, \quad b_{i,k} = \frac{\tilde{f}_{\text{int}}(\Phi(p_i), U_i q^{(i,k)})}{\|\tilde{f}_{\text{int}}(\Phi(p_i), U_i q^{(i,k)})\|}. \tag{18}$$

Here, $U_i^e \in \mathbb{R}^{12\times r}$ and $f_{\text{int}}^e \in \mathbb{R}^{12}$ are the basis U_i and internal forces restricted to element e, respectively.

After $\mathcal{C}$ is determined, we then separately optimize the cubature weights v_i for each pose, under the element set $\mathcal{C}$, by solving

$$\begin{bmatrix} g_i^1 & g_i^2 & \cdots & g_i^c \end{bmatrix} v_i = b_i, \quad v_i \geq 0. \tag{19}$$

Such an additional optimization step gives per-pose optimized cubature weights, and lowers the relative error at each pose. At runtime, we obtain the interpolated cubature weights $v(p)$ as

$$v(p) = \sum_{i=1}^{m} w_i(p) v_i. \tag{20}$$

The reduced internal forces $\tilde{f}_{\text{int}}(\Phi(p),U(p)q)$ and stiffness matrix $\tilde{K}(\Phi(p),U(p)q)$ can be approximated as

$$\tilde{f}_{\text{int}}\Big(\Phi(p),U(p)q\Big) = \sum_{e\in\mathcal{C}} v^e(p) U^e(p)^T f_{\text{int}}^e\Big(\Phi(p),U^e(p)q\Big) \tag{21}$$

$$\tilde{K}\Big(\Phi(p),U(p)q\Big) = \sum_{e\in\mathcal{C}} v^e(p) U^e(p)^T K^e\Big(\Phi(p),U^e(p)q\Big) U^e(p). \tag{22}$$

At runtime, we evaluate f_{int}^e and K^e with respect to the *exact* rest shape positions $\Phi(p)$. We do so by explicitly computing the rest positions, in pose p, of the cubature tets $\mathcal{C}$ only (Section 4.1). Because the set $\mathcal{C}$ represents only a small fraction of the entire tetrahedral mesh, these rest shapes can be computed with minimal overhead.

5.4 Pose-Space Cubature for Inertial Forces

If $\bar{\Phi}$ is explicitly described, we can find a closed-form expression for $U^T f_{\text{inert}}$. For a general $\bar{\Phi}$, one approach is to evaluate f_{inert} using finite differences, and then project. Such a calculation, however, involves full-space quantities. We now give an approach that computes $\tilde{f}_{\text{inert}}$ using cubature approximation for a general "black-box" $\bar{\Phi}$. The goal of the training stage is to locate a small cubature set of vertices $\mathcal{C}'$ such that

$$\tilde{f}_{\text{inert}} = \sum_{s\in\mathcal{C}'} v'^s U^s(p)^T f_{\text{inert}}^s \tag{23}$$

approximates reduced inertial forces to some desired degree of accuracy. Here v' are the cubature weights, and $U^s(p) \in \mathbb{R}^{3\times r}$ and f_{inert}^s are the basis $U(p)$ and inertial force f_{inert} restricted to vertex s, respectively. We evaluate it as

$$f_{\text{inert}}^s = \sum_j m_{sj} \frac{d^2 x_j}{dt^2}, \tag{24}$$

where x_j is the j-th vertex entry of $\Phi(p)$. The summation runs over all vertices of tets adjacent to vertex s, and m_{sj} is the 3×3 block of the mass matrix corresponding to vertices s and j. We evaluate the acceleration of vertex j using finite differences. Therefore, at run time, we only need to evaluate accelerations on a small set of tet mesh vertices.

Cubature training: To determine $\mathcal{C}'$ and v', we first randomly perturb the pose p around each pose p_i, generating $\mathcal{T}'+2$ sample deformations of the tet mesh for each i. Our perturbations have a prescribed maximum magnitude: we use 1° for rotation angles, and $1mm$ for translational degrees of freedom of human-sized characters. We treat the $\mathcal{T}'+2$ samples as consecutive frames in time and use them to obtain $\mathcal{T}'$ sample inertial forces with finite differences. We then combine all the samples at all poses into a global cubature solve. Namely, we find $\mathcal{C}'$ and v' using Equation 17, where the k-th component of vectors $g_i^s \in \mathbb{R}^{\mathcal{T}'}$ and $b_i \in \mathbb{R}^{\mathcal{T}'}$ for $k = 1,\ldots,\mathcal{T}'$, is

$$g_{i,k}^s = \frac{U_i^{sT}(f_{\text{inert}}^s(x_i)^{(k)})}{\|U_i^T(f_{\text{inert}}(x_i)^{(k)})\|}, \quad b_{i,k} = \frac{U_i^T(f_{\text{inert}}(x_i)^{(k)})}{\|U_i^T(f_{\text{inert}}(x_i)^{(k)})\|}. \tag{25}$$

Here, $s = 1,\ldots,|\mathcal{C}'|$ denotes the s-th cubature tetrahedral mesh vertex. We could continue optimizing the cubature weights for each pose with pose-space cubature, analogous to Equation 19. However, we found that the globally obtained $\mathcal{C}'$ and v' already work well in practice. Our relative training error is under 3% both for elasticity and inertial force, in all examples.

6 Self-Contact-Aware Model Reduction

Self-collision detection and response are expensive operations for real-time applications. Even if they were inexpensive, the dynamics in the presence of contact will be incorrect if the basis does not incorporate contact. We first resolve self-contact at each pose using any existing method. This can be done by the artist when sculpting a PSD pose, or it can be automated using a self-collision resolution algorithm. The contact corrections are simply stored into the PSD correction vector $\bar{\delta}(p)$. Next, we construct a basis that is aware of the contact constraint, and as such the modes prohibit any deeper penetrations (Figure 5). This is done by solving a *constrained sparse generalized eigenvalue problem*, for which we propose an efficient numerical procedure. To the best of our knowledge, we are first to incorporate such pose-dependent contact into the construction of the basis for model reduction, by giving a practical algorithm to compute linear vibration modes under constraints. Although the contact is a unilateral constraint, we choose to model it as a bilateral constraint that keeps the contact in place. This approximation is motivated by two practical observations: (1) self-contact that we are interested in occurs near the joints, such as the elbow and the hip. In poses p that are self-colliding, assuming p is held fixed, the contacting elastic material is in permanent contact as the two sides of the arm or leg are firmly pressing against each other. Hence, it is difficult for such a contact to separate, but a deeper penetration could occur without a bilateral constraint, and (2) we are targeting interactive applications where accuracy can be traded for speed.

Our contacts consist of c contact vertices of the tet mesh Γ, along with a contact normal N_i, $i = 1,\ldots,c$. The contacts are detected by running collision detection on the surface mesh of the fitted tet mesh for each pose. Bilateral contact constraints can then be expressed as $Cu = 0$, where $C \in \mathbb{R}^{c\times 3n}$. Our constraints include fixed constraints, $S_i u = 0$, or only permit sliding motion in the plane normal to contact, $N_i^T S_i u = 0$, where $S_i \in \mathbb{R}^{3\times 3n}$ is the matrix that selects the DOFs of the contact vertex from displacement u of Γ. We always enforce full-rank property of C, by removing redundant rows using SVD, i.e., removing all the corresponding right singular rows of C for singular values under some threshold ε; we use $\varepsilon = 10^{-6}$.

Given the constraint matrix C, we compute the linear modes $\varphi_i \in \mathbb{R}^{3n}$ as the smallest $\hat{r}$ eigenvectors of the constrained generalized eigenproblem

$$K\varphi_i = \lambda_i M\varphi_i, \qquad \text{subject to } C\varphi_i = 0. \tag{26}$$

The modes can be obtained by finding the nullspace $\mathcal{Z} \in \mathbb{R}^{3n\times(3n-c)}$ of C, and then set $\varphi_i = \mathcal{Z}y_i$, for some unknown $y_i \in \mathbb{R}^{3n-c}$. The constrained eigenproblem becomes an unconstrained eigenproblem,

$$\left(\mathcal{Z}^T K\mathcal{Z}\right)y_i = \lambda_i\left(\mathcal{Z}^T M\mathcal{Z}\right)y_i. \tag{27}$$

Because K and M are symmetric positive-definite, the matrices $\hat{K} = \left(\mathcal{Z}^T K\mathcal{Z}\right)$ and $\hat{M} = \left(\mathcal{Z}^T M\mathcal{Z}\right)$ are also symmetric and positive-definite (proof in Appendix A). Hence, the Eigenproblem 27 can be solved using a standard generalized Arnoldi eigensolver [Lehoucq et al. 1997]. Special care needs to be taken because the matrices $\hat{K}$ and $\hat{M}$ are obtained by multiplying sparse matrices, but are not themselves sparse. We address this issue by noting that the Arnoldi-based solvers only require a "black-box" ability to multiply an arbitrary vector x with the matrix $\hat{M}^{-1}\hat{K}$. Multiplying $\hat{K}x$ can be performed efficiently simply by multiplying with sparse matrices $\mathcal{Z}^T, K, \mathcal{Z}$. Multiplying with $\hat{M}^{-1}$ can be performed by solving a linear system using the conjugate gradient method, where only multiplications with $\hat{M}$ are required. We augment the linear modal basis with constrained modal derivatives ψ_{ij} by solving

$$\left(\mathcal{Z}^T K\mathcal{Z}\right)z_{ij} = -\mathcal{Z}^T\left((H:\varphi_i)\varphi_j\right), \quad \psi_{ij} = \mathcal{Z}z_{ij}, \tag{28}$$

where H is the Hessian stiffness tensor. Again, the linear system can be solved using conjugate gradients. Only multiplications with sparse matrices $\mathcal{Z}^T$, K and $\mathcal{Z}$ are required.

We find the nullspace $\mathcal{Z}$ as follows. Because C is a full-rank matrix, the fundamental theorem of linear algebra guarantees that there exists a permutation matrix $P \in \mathbb{R}^{3n\times 3n}$ that permutes the columns of C so that the first c columns of $CP = \begin{bmatrix} C_p & C_n \end{bmatrix}$ form an invertible matrix $C_p \in \mathbb{R}^{c\times c}$, and $C_n \in \mathbb{R}^{c\times(3n-c)}$. The nullspace matrix can then be obtained as

$$\mathcal{Z} = P\begin{bmatrix} -C_p^{-1}C_n \\ I \end{bmatrix}. \tag{29}$$

The process of finding the permutation matrix is equivalent to LU factorization with column pivoting, $CP = L\begin{bmatrix} U_p & U_n \end{bmatrix}$, where $L \in \mathbb{R}^{c\times c}$ is lower-triangular, $U_p \in \mathbb{R}^{c\times c}$ is upper-triangular, and $U_n \in \mathbb{R}^{c\times(3n-c)}$. After performing the LU factorization with column pivoting, we then compute and store the top-block of $\mathcal{Z}$ as

$$-C_p^{-1}C_n = (LU_p)^{-1}(LU_n) = U_p^{-1}U_n. \tag{30}$$

Instead of using conjugate gradients, one can solve linear systems with $\hat{M}$ and $\hat{K}$ approximately, by cutting all entries of $\mathcal{Z}$ with an absolute value below a small threshold $\varepsilon = 10^{-10}$. This converts $\mathcal{Z}$, and therefore $\hat{M}$ and $\hat{K}$, into sparse matrices, and one can then use a direct linear system solver (we use Pardiso [Pardiso 2015]). We accelerate the CG solver by using the thresholded solver as a preconditioner, which gave us a speedup of approximately $1.5\times$ in our examples. Our preconditioned CG method computed the constrained eigenmodes in a few seconds (Table 1), making it possible to easily generate constrained modes in our preprocessing pipeline.

	$3n$	c	uncons	cons-thresh	cons-CG
human arm	7092	54	0.9 s	1.2 s	2.6 s
polar bear	12054	73	1.9 s	2.2 s	7.3 s
gorilla	11313	114	1.7 s	2.4 s	6.2 s

Table 1: *Eigenmodes computation (representative pose, 20 modes): unconstrained (uncons), constrained thresholded (cons-thresh), constrained using CG (cons-CG; preconditioned with cons-thresh). Scalability with #modes (gorilla): cons-CG takes 5.4s, 6.2s, 11.7s, 18.7s for 10, 20, 50, 100 modes, respectively.*

7 Localized Model Reduction

With spatially global modal vectors, the complexity of runtime basis interpolation and cubature-based reduced simulation grow linearly and cubically with the number of modal vectors r, respectively. Although spatially global basis vectors produce good results for small and moderate problems, local deformation handling is generally improved by increasing r. We address this problem by computing localized basis functions. We do so by computing smooth scalar weights that define overlapping local regions of the tetrahedral mesh (Figure 2 (a)), and then compute a localized basis in each region. We note that it has been challenging to define localized basis functions for model reduction that can work with large deformations induced by a character rig. One proposed solution has been to partition the character into multiple domains, model-reduce each domain, drive the rigid body motion of each domain by a skeleton, and employ coupling elastic forces at the joints to keep the deformation continuous [Kim and James 2012]. Because our pose-space basis construction does not need joints, our local regions do not need to be partitioned at the joints, and can (and should) overlap partially with each other. We show how to construct basis functions that decay arbitrarily smoothly at the boundary of each region. As a result, there are no seam artifacts at the

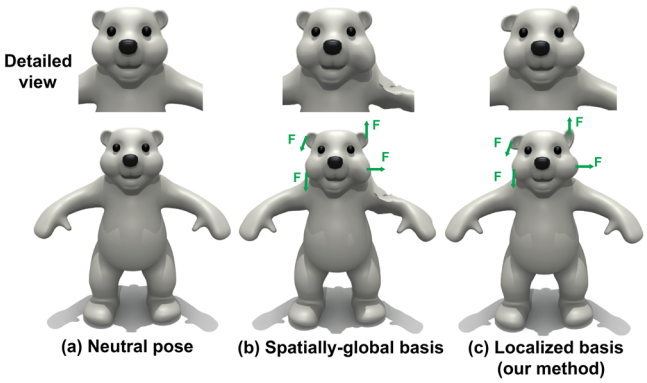

Figure 11: Our method supports local deformations. *Our localized basis (*20 *modes on the head, and* $r = 100$ *total for the entire model) produces good local deformations (c). A basis of global methods requires* 180 *modes to produce motion in the bear's cheeks, whereas the ears still have no local deformations. Furthermore, the globality of the global modes causes "modal ringing": visible deformations appear in the bear's left arm when pulling the cheeks (b), whereas our method has no such artifact (c).*

region boundaries. Combined with our pose-space basis construction, this gives us hard real-time character soft-body dynamics that exhibits localized deformations, free of any pose-interpolation or basis spatial discontinuity artifacts.

The user first selects $d \geq 1$ control handles (points or line segments), to denote d individual regions $\mathcal{D}$. We then compute smooth, localized scalar bi-harmonic weight functions $\mathcal{W}_i \in \mathbb{R}^n$ on the vertices of the tetrahedral mesh [Jacobson et al. 2011], under the condition that the weight is non-negative everywhere, equals 1 at the i-th control handle, and is 0 at all the other handles. We place a vertex to region $\mathcal{D}_i$ if the vertex weight in $\mathcal{W}_i$ is larger than a threshold η; we use $\eta = 0.1$. After thresholding, we remap $\mathcal{W}_i$ from $[\eta, 1]$ to $[0,1]$ such that the weights smoothly decay to 0 on the region boundary, by introducing $\mathcal{W}_i' = (\mathcal{W}_i - \eta)^\kappa / (1-\eta)^\kappa$, where $\kappa > 0$ is a scalar parameter that controls the degree of locality and how rapidly the weights decay to zero at the boundary; we use $\kappa = 1/2$.

In each region $\mathcal{D}_i$, we compute a localized basis by fixing the tet mesh vertices outside of $\mathcal{D}_i$, and progressively increasing the material stiffness, based on $\mathcal{W}_i'$, as we approach to the region boundary. The linear modes for a mesh with such modified material properties can be computed by solving the generalized eigenvalue problem

$$W^{-T} K W^{-1} \varphi_i = \lambda_i M \varphi_i, \tag{31}$$

where $W = W^T \in \mathbb{R}^{3n \times 3n}$ is a diagonal matrix consisting of $\mathcal{W}_i'$. Note that Because the stiffness matrix K is inversely weighted with the scalar weight function, the modes are biased towards the parts that have higher weights. Therefore, the deformation decays smoothly to 0 on the boundary of each region. Equation 31 is equivalent to

$$K y_i = \lambda_i W^T M W y_i, \quad \varphi_i = W y_i, \tag{32}$$

and therefore the numerical problem of inverting W is avoided. Analogously, the modal derivatives are localized by solving

$$K z_{ij} = -(H : \varphi_i) y_j, \quad \psi_{ij} = W z_{ij}. \tag{33}$$

Our examples use localized modal analysis and self-contact-aware model reduction simultaneously. The linear modes and modal

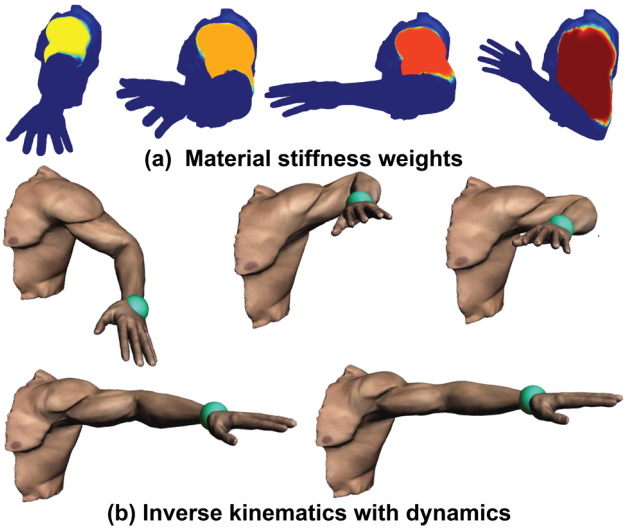

Figure 12: Real-time inverse kinematics with FEM dynamics: *The human arm (*4 *rig DOFs) is driven in real-time by inverse kinematics (IK handle is shown in green). Our method computes the dynamics caused by inertial forces, in hard real time (*1,100 *FPS). The material stiffness of the upper arm muscle is designed to vary at each pose (a), and this is incorporated into the basis at each pose. Our method interpolates the materials in pose space at runtime.*

derivatives are obtained by solving

$$(\mathcal{Z}^T K \mathcal{Z}) y_i = \lambda_i (\mathcal{Z}^T W^T M W \mathcal{Z}) y_i, \quad \varphi_i = W \mathcal{Z} y_i, \tag{34}$$

$$(\mathcal{Z}^T K \mathcal{Z}) z_{ij} = -\mathcal{Z}^T \Big((H : \varphi_i) \mathcal{Z} y_j \Big), \quad \psi_{ij} = W \mathcal{Z} z_{ij}, \tag{35}$$

where $\mathcal{Z}$ is formed using the weighted constraint matrix CW.

Our basis vectors are spatially sparse and we exploit the sparsity by storing them as compressed matrices. Basis interpolation, reduced internal force, tangent stiffness matrix, inertial force computation and deformation vector assembly all operate only on non-zero entries. There are no other changes needed to our system. The regions are kept the same for all the poses p_i, but the localized basis vectors are different in each pose, due to the modified rest configuration geometry in each pose. We demonstrate locality in Figure 11.

8 Pose-Space Materials

So far, we have assumed that the material properties are constant throughout our tetrahedral mesh. However, as the pose p changes, not only the rest shape $\Phi(p)$ changes, but the elastic material properties may also change. As an example, the muscle region of a rigged arm should be stiffer when the arm bends and the muscle contracts (Figure 12, (a)). In principle, the degrees of freedom of material design for the artist are three-fold: material space (the specific shape of the strain-space curve), spatial space (materials that vary across the mesh) and pose-space (pose-dependent material). In this paper, we do not investigate material space design (for a recent approach, see [Xu et al. 2015b]). We enable spatial and pose-space design by letting the artist paint the spatial material E, optionally in each pose, $E(p_i)$. We then construct the pose-dependent basis $U(p_i)$ using $E(p_i)$. We note that per-pose materials have been previously explored by [Galoppo et al. 2009], who performed the simulation using St.Venant-Kirchhoff material in a pose-independent basis U, whereas we simulate general nonlinear materials in a pose-specific basis. Our cubature training at each pose, of course, also needs to incorporate the specific material at the pose. At runtime,

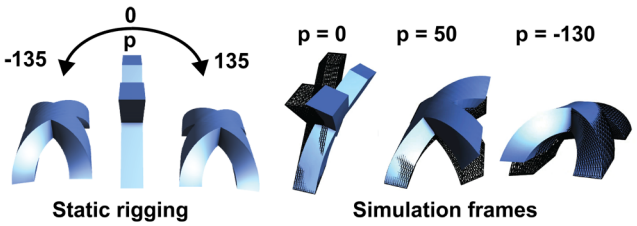

Figure 13: The "X". *Maya nonlinear deformer (1-DOF rig function); three poses. Rest shapes are shown in wireframe. Reduced dynamics are driven at* 1,700 *FPS by inertial and external forces.*

we simply interpolate the material on the cubature elements $\mathcal{C}$ as

$$E(p) = w(p)[E(p_1), E(p_2), \ldots, E(p_m)]^T. \tag{36}$$

The rest of our system remains the same as with constant materials. Therefore, material control can be easily integrated into our system, and adds negligible runtime computational or memory overhead.

In our system, the artist paints the material properties $E(p_i)$ by specifying a (heterogeneous) scalar field of Young's modulus or Poisson's ratios. The artist can paint them directly on the tetrahedral mesh in each pose, as computed in Section 4.1. Alternatively, the artist can paint them on the triangle mesh. We then need to transfer them to the tetrahedral mesh. Similar to tetrahedral mesh fitting, we do this by solving a least square fitting problem

$$E(p_i) = \arg\min_{E} \sum_{e=1}^{T} \left(\|E^e - \bar{E}^e\| \right) + \zeta \frac{1}{2} E^T L E, \tag{37}$$

where T is the number of tetrahedral mesh elements, $\bar{E}^e$ is the average painted value specified among the triangle mesh vertices inside the element e, $\zeta > 0$ controls smoothness of E, and $L \in \mathbb{R}^{T \times T}$ is the tet mesh volume-weighted Laplacian matrix [Xu et al. 2015a].

9 Results

In the human arm example (StVK material, Figure 12), we rig the arm with 3 skeleton bones. We use the 3 rotational DOFs of the shoulder joint and 1 rotational DOF (yaw) of the elbow joint. All the tets intersecting the skeleton are fixed. We engaged an artist to sculpt 10 poses (Figure 9) to resolve skinning artifacts and self-collisions between the forearm, upper arm and the body. The material distribution is heterogeneous and varies in different poses. We define 3 local regions, each of which has 15 modes, for a total of 45 modes. We drive the human arm with an inverse kinematic handle, solving in real-time for the joint angles, which are then sent to our system as input. Reduced simulation runs at 1,100 FPS, producing soft-tissue dynamics originating from the inertial forces.

In our second example (neo-Hookean material, Figure 13), we deform the X with a Maya nonlinear bending deformer. The rig motion is controlled by a 1-DOF rig parameter, i.e., the bending curvature, which is interactively adjusted by the mouse. We choose the neutral shape and the two rigged shapes at $p = -135$ and $p = 135$ as poses, and compute 30 modes at each pose. The geometry changes vastly in this example and our interpolated subspace produces natural motions in all the states (1,700 FPS).

Our third example (StVK material, Figure 2) is a polar bear rigged with a Maya wire deformer (144 rigging DOFs). Animation sequences were created by an artist, by keyframing the rig motion. We picked 8 representative poses, sculpted PSD corrections, and addressed the self-contacts. Our constrained basis makes it possible to have a contact-free motion without performing run-time self-collision detection. Localized basis with 6 regions ($r = 15$ for arms and the legs, and $r = 20$ for the head and the body, 100 modes total) produces rich localized and global dynamics (700 FPS), without the need for a large set of spatially global modes (Figure 11). By using only 50 and 25 linear modes, we can increase the performance to 1250 and 1800 FPS, respectively, at only slightly decreased quality.

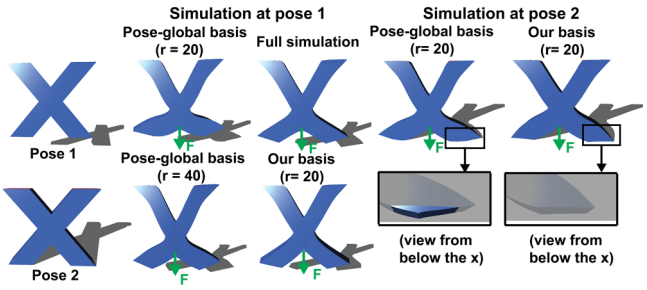

Figure 14: Comparison to pose-independent PCA basis. *Two poses. The "X" is permanently fixed at the top. In pose* 1*, it is in the air and free of ground contact. In pose* 2*, the bottom is in contact with the ground plane and downward motion is restricted by our contact-aware modal basis. We compute* 20 *modes at each pose. For comparison, pose-global modes are computed by performing PCA on the union of our modes computed at the two poses. At pose* 1*,* 20 *fixed PCA modes produce unnatural motion, as compared to the full simulation ground truth, whereas our method matches the ground truth much more closely. To achieve similar quality as our method that only has* 20 *modes, $r = 40$ pose-independent modes are needed. At pose* 2*, our method correctly handles contact. The pose-independent modes violate contact constraint easily since they mix content from pose 1 and pose 2, and therefore some of the unwanted degrees of freedom for downward motion creep into the basis.*

The gorilla example (StVK material, Figure 1) is rigged with 16 skeleton joints. We use the translational and rotational DOFs for 10 joints (60 rig DOFs). We selected 4 typical poses. Localized subspaces are computed for 6 regions ($r = 10$ for the two legs, and $r = 20$ for the head, body and two arms, a total of 100 modes). We obtain the input gorilla skeletal motion by tracking a human with Kinect 1.0, and then retarget the human skeleton to the gorilla in real-time using the SmartBody system [Feng et al. 2015]. Our method produces quality secondary dynamics at 750 FPS. Model reduction has the nice property that it is easy to adjust the trade-off between quality and speed, simply by altering the number of modes [James and Pai 2002]. By using only 50; 25; 10 modes total, we can accelerate the multi-core performance to 1550; 2200; 2350 FPS, respectively, at a small loss of quality (e.g., in the gorilla's chin and ears, visible in the supplemental video). The reason for why $r = 25$ and $r = 10$ have similar performance is because of the overhead of multi-threading, and because the rt:core and rt:render costs of Table 3 become dominant. The single-core performances are 240; 600; 1400; 2200 FPS for $r = 100; 50; 25; 10$, respectively.

We give a detailed breakdown on the theoretical and practical performance and memory costs of our method in Table 2. Overall measured performance is given in Table 3. We timestep the reduced dynamics using the implicit backward Euler integrator; dense system solves are performed using Intel MKL. The time cost for the other parts, such as cubature weight interpolation and radial basis weights evaluation, is negligible. Because we are using the cubature approximation for the reduced inertial force, our system does not require evaluating/fitting the position $\Phi(p)$ of all tet mesh vertices at each runtime frame. Instead, the tetrahedral mesh vertex positions are only required for the vertices of the internal force cubature elements $\mathcal{C}$, and reduced force cubature vertices $\mathcal{C}'$ and their neighbors. Even for (more seldomly used) rig functions that cannot compute a small number of vertices much faster than all of the ver-

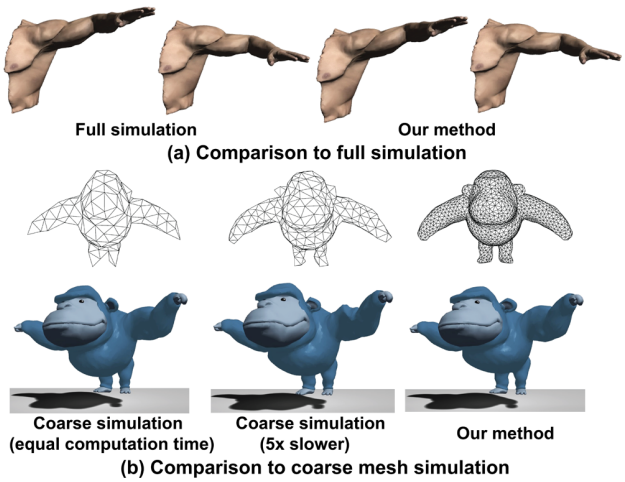

Figure 15: Comparison to full simulation and coarse mesh simulation. *(a) Our method produces a good approximation to the full simulation while being* $60\times$ *faster. (b) We compared our method to coarse mesh full simulation. We adjusted the coarse mesh resolution so that the computation time matches that of our method (first column). The tetrahedral mesh only has* 169 *vertices and* 420 *tets. A large percentage of tets intersect with the bones and are kept fixed. The arms, hips and the legs does not have any dynamics. In the second column, we create a more detailed, but still coarse, mesh (*417 *vertices,* 1229 *tets). Now, our method is* $5\times$ *faster, and still produces better dynamics. While more tets are free in the coarse mesh, the hips and the legs are still very rigid, and the dynamics overall is low quality (such as the arms and the belly). Our method is faster and produces quality dynamics.*

tices, cubature approximation avoids full-space operations, such as computing the full-space internal and inertial forces and their subspace projections. While we do not completely remove all the full-mesh operations, our simulation is very amenable to parallelization on the CPU or GPU. We render our dynamically deforming models in real-time at 90 FPS using shader-based OpenGL, including real-time shadows, self-shadows and ambient occlusion. We take a single simulation step per graphics frame in all examples; i.e., our simulation timestep is $1.0/90s$. We note that this is a general good property of model reduction: it filters high-frequency elasticity content and hence improves stability, or permits larger timesteps.

Previous methods constructed a pose-independent basis using PCA [Galoppo et al. 2009; Teng et al. 2015]. However, the modes change substantially with the poses, due to the changing geometry, material properties and contact conditions. Therefore, the pose-independent basis will need a large set of basis vectors to have quality dynamics globally in the pose space (Figure 14, left). Pose-independent basis PCA mixes the modes computed under different contact conditions. Therefore, contact constraints can be easily violated (Figure 14, right). Our method always uses a much more compact set of basis vectors, specific to each individual shape. Furthermore, the pose-independent basis methods transform the deformations back to the neutral pose and simulate the dynamics around the neutral pose. This will lead to incorrect dynamics, because the force models at different poses vary substantially due to geometry and material changes (Figure 4, Section 8).

We also compare our basis interpolation method to the adaptive subspace method [Hahn et al. 2014], which reassembles the reduced system at each timestep, i.e., computes $U^T KU, U^T MU, U^T f$. These projections have a high total running time of $O(nr^2)$, and take up a significant fraction of the computation time. In our system, assembling the reduced system does not depend on the mesh resolution. With equal subspace dimension, our reduced system assembly and basis interpolation is two orders of magnitude faster than their reduced system assembly: human arm: 0.4 msec vs 38 msec ($95\times$), polar bear: 0.95 msec vs 82 msec ($86\times$), gorilla 1.0 msec vs 320 msec ($320\times$). Even when we count the entire simulation costs of one timestep of our method, our simulation is $44\times, 57\times, 240\times$ faster than their subspace assembly costs alone in the three respective examples (and hence even faster against their entire costs). A similar argument applies to the fast rig-space physics method [Hahn et al. 2013], which also needs to construct $J^T KJ, J^T MJ, J^T f$ at every timestep, where J is their rig-Jacobian function.

	time [msec]		storage [MB]	
pose-space deformation	$O(\bar{n})$	0.07	$3\bar{n}m$	0.4
tetrahedral mesh fitting	$O(d)$	0.12	$3md$	3
basis interpolation	$O(nr)$	0.2	$3nrm$	4.3
reduced internal forces	$O(r^3)$	0.48	$2\lvert\mathcal{C}\rvert$	0.01
reduced inertial forces	$O(r^2)$	0.11	$2\lvert\mathcal{C}'\rvert$	0.01
reduced integration	$O(r^3)$	0.24	0	0
construct $\bar{u} = AU(p)q$	$O(nr+\bar{n})$	0.1	$4\bar{n}$	0.2

Table 2: *Performance and storage complexity (gorilla,* $r = 100$*);* $d = |\mathcal{C}| + |neigh(\mathcal{C}')|$*. The PSD storage is conservative because in practice PSD corrections are only non-zero in the vicinity of the joints. We always perform the basis interpolation on the full mesh. Because rendering needs the full mesh basis, whereas simulation only needs the basis at d vertices, further speedups would be possible if multiple simulation steps are taken per rendering frame. Reduced internal force row includes the reduced stiffness matrix. The storage for displacement vector assembly and interpolation (last row) consists of the interpolation weights between the tetrahedral and triangle mesh (matrix A).*

Our reduced dynamics provides a good approximation to full simulation (Figure 15, (a)), while being 40-125$\times$ faster than full simulation (Table 3). Note that here, full simulation is allowed to use our fast tetrahedral mesh fitting $\Phi(p)$. Without our fast fitting, with the full simulation fitting a "physical" tet mesh at each frame [Barbič et al. 2009], full simulation is $2,000\times$ slower than our method. A natural question to ask is whether results similar to ours in terms of both speed and quality could be achieved by running a full simulation [Capell et al. 2005; McAdams et al. 2011; Kim and Pollard 2011; Liu et al. 2013] that is sufficiently coarse to achieve the needed speed. We performed such an experiment and report the results in Figure 15, (b). Under equal computation time, the coarse mesh is overconstrained to the skeleton, and does not have enough degrees of freedom, producing very suboptimal dynamics. Even when a finer coarse mesh is used where overconstraining is no longer a substantial issue, our method is $5\times$ faster, and produces a much higher quality dynamics.

10 Conclusion

We presented a method to add physically based dynamics to pose-space deformation and character rigging. Our method augments pose-space deformation with high-quality secondary soft-tissue dynamics under arbitrary rigged or skeletal motion. It runs at milliseconds per frame for complex characters, presents only a minimal change to the existing character animation pipelines, is artist-directable through painting of stiffness maps, supports localized basis functions and respects skin contacts.

We treat the entire object as one mesh, optionally with non-homogeneous material properties, but do not simulate the internal anatomical structure such as bones, muscles and tendons. While we use standard pose-space deformation, our system is orthogonal

Example	tet-vtx	tet-el	tri-vtx	r	reg	poses	PSD	cub:int	cub:inert	rt:core	rt:dyn	rt:render	rt:total	mem	FPS	sp
arm	2364	8122	10525	45	3	10	9	192	45	36 %	58 %	6%	0.87 ms	11 MB	1150	60×
X-shape	3077	9303	2362	30	1	3	0	144	50	26 %	65 %	9%	0.59 ms	4.2 MB	1700	105×
polar bear	4018	12762	6876	100	6	8	7	336	200	29 %	66 %	5%	1.43 ms	7.3 MB	700	40×
gorilla	3771	16297	15774	100	6	4	2	144	119	29 %	63 %	8%	1.33 ms	7.9 MB	750	125×

Table 3: Simulation statistics *for #tet mesh Γ and triangle mesh $\bar{\Gamma}$ vertices and elements (tet-vtx, tet-el, tri-vtx), #of reduced DOFs (r), #local regions (reg), #poses with a simulation basis (poses) and PSD-corrected geometry (PSD), #cubature samples for internal and inertial forces (cub:int, cub:inert), total memory (mem), frame rate (FPS), speedup (sp) over full simulation, simulation step time (rt:total), and its breakdown in terms of timestepping reduced dynamics (rt:dyn: reduced internal and inertial forces, and integration), constructing deformations for rendering (rt:render, $\bar{u} = AU(p)q$), and the rest (rt:core; including sparse tet mesh fitting and basis interpolation). We use* 8 *threads with OpenMP; no GPU computation. Intel Xeon* 2.9 *GHz CPU (*2×8 *cores;* 32*GB RAM); GeForce GTX* 680 *graphics card (*2*GB RAM).*

to other more advanced pose-space deformation techniques, such as weighted pose-space deformation [Kurihara and Miyata 2004] and weight-decay regularization [Anjyo et al. 2014]. For shapes far from training data, the normalized biharmonic RBF weights used in our system converge to a constant value (a limit, independent of the radial direction) that is a convex combinations (with weights on $[0,1]$) of the artist-sculpted PSD corrections. It would be interesting to explore such and other advanced sparse data interpolation techniques in the future. Our collisions are treated as bilateral constraints in each pose. Although unilateral contact would be in principle more desirable, our approach is fast, and prevents our dynamics from causing self-collisions. In typical poses where character self-collisions occur, such as bent elbows or shoulders, the self-colliding surfaces are pushing against each other due to contact. It is therefore difficult for the skin to break contact anyway without modifying the pose. If the pose itself breaks contact, so does our method. Hence, treating the contact bilaterally does not significantly decrease realism in these typical scenarios. For poses close to contact, the interpolated basis can be contaminated by the contact-aware basis computed at the contact pose. While this effect is small and we did not observe artifacts in our examples, the problem could be alleviated by adding a near-contact pose into the set of poses with a reduced model. Our method does not prevent self-collisions that occur because the input rigged animation causes the character to self-collide. These self-collisions must be avoided in another way, such as by properly modifying the skeleton joints or rig parameters. We believe these compromises are reasonable for applications in games and virtual reality, where speed is paramount. In the future, we would like to investigate how to add anatomy to our simulations, or incorporate data scanned from real subjects. For example, as opposed to computing the basis using linear modes and derivatives, we could form our basis by performing PCA on the anatomically simulated, or scanned data, in the vicinity of each pose. We did not use GPU computation; several stages of our pipeline could be significantly accelerated on the GPU.

Acknowledgements

This research was sponsored in part by the National Science Foundation (CAREER-1055035, IIS-1422869), the Sloan Foundation, the Okawa Foundation, and USC Annenberg Graduate Fellowship to Hongyi Xu. We would like to thank Dr. Ari Shapiro for help with the SmartBody system, Carlos Joy for modeling, and Bohan Wang for his real-time OpenGL renderer.

A Proof that $\mathcal{Z}^T A \mathcal{Z}$ is SPD

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive-definite (SPD) matrix, and $\mathcal{Z} \in \mathbb{R}^{n \times k}$ be a full-rank matrix, for $k \leq n$. $\mathcal{Z}^T A \mathcal{Z}$ is obviously symmetric. Suppose $x^T \left(\mathcal{Z}^T A \mathcal{Z}\right) x = 0$ for some vector $x \in \mathbb{R}^k$. Then, $y^T A y = 0$, for $y = \mathcal{Z} x$. Because A is SPD, we must have $y = 0$, and therefore, as $\mathcal{Z}$ is full-rank, we have $x = 0$. Hence, $\mathcal{Z}^T A \mathcal{Z}$ is SPD.

References

AMSALLEM, D., AND FARHAT, C. 2008. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA J. 46*, 7, 1803–1813.

AMSALLEM, D., AND FARHAT, C. 2011. An online method for interpolating linear parametric reduced-order models. *SIAM J. on Scientific Computing 33*, 5, 2169–2198.

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. on Graphics (SIGGRAPH Asia 2008) 27*, 5, 165:1–165:10.

ANJYO, K., LEWIS, J. P., AND PIGHIN, F. 2014. Scattered data interpolation for computer graphics. In *SIGGRAPH Courses*, 27.

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

BARBIČ, J., AND ZHAO, Y. 2011. Real-time large-deformation substructuring. *ACM Trans. on Graphics (SIGGRAPH 2011) 30*, 4, 91:1–91:7.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. *ACM Trans. on Graphics (SIGGRAPH 2009) 28*, 3.

BARNHILL, R., DUBE, R., AND LITTLE, F. 1983. Properties of Shepards Surfaces. *Rocky Mountain J. Math. 13*, 365–382.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM Trans. on Graphics (SIGGRAPH 2002) 21*, 3, 586–593.

CAPELL, S., BURKHART, M., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2005. Physically based rigging for deformable characters. In *Symp. on Computer Animation (SCA)*, 301–310.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *Proc. of ACM SIGGRAPH 2001*, 67–76.

CHEN, D., AND PLEMMONS, R. J. 2006. Nonnegativity constraints in numerical analysis. In *Symp. on the Birth of Numerical Analysis*.

DEGROOTE, J., VIERENDEELS, J., AND WILLCOX, K. 2010. Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis. *Int. J. for Numerical Methods in Fluids 63*, 2, 207–230.

FENG, A., CASAS, D., AND SHAPIRO, A. 2015. Avatar reshaping and automatic rigging using a deformable model. In *Proc. of the Motion in Games (MIG) Conf.*, 57–64.

GALOPPO, N., OTADUY, M. A., TEKIN, S., GROSS, M., AND LIN, M. C. 2007. Soft articulated characters with fast contact handling. In *Computer Graphics Forum*, vol. 26, 243–253.

GALOPPO, N., OTADUY, M. A., MOSS, W., SEWALL, J., CURTIS, S., AND LIN, M. C. 2009. Controlling deformable material with dynamic morph targets. In *Proc. of the Symp. on Interactive 3D graphics and games (I3D)*, ACM, 39–47.

GOWER, J. C., AND DIJKSTERHUIS, G. B. 2004. *Procrustes problems*, vol. 3. Oxford University Press Oxford.

HAHN, F., MARTIN, S., THOMASZEWSKI, B., SUMNER, R., COROS, S., AND GROSS, M. 2012. Rig-space physics. *ACM Trans. on Graphics (SIGGRAPH 2012) 31*, 4, 72:1–72:8.

HAHN, F., THOMASZEWSKI, B., COROS, S., SUMNER, R., AND GROSS, M. 2013. Efficient simulation of secondary motion in rig-space. In *Symp. on Computer Animation (SCA)*, 165–171.

HAHN, F., THOMASZEWSKI, B., COROS, S., SUMNER, R. W., COLE, F., MEYER, M., DEROSE, T., AND GROSS, M. 2014. Subspace clothing simulation using adaptive bases. *ACM Trans. on Graphics (SIGGRAPH 2014) 33*, 4, 105:1–105:9.

HARMON, D., AND ZORIN, D. 2013. Subspace integration with local deformations. *ACM Trans. on Graphics (SIGGRAPH 2013) 32*, 4, 107:1–107:9.

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

HILDEBRANDT, K., SCHULZ, C., VON TYCOWICZ, C., AND POLTHIER, K. 2012. Interactive spacetime control of deformable objects. *ACM Trans. on Graphics (SIGGRAPH 2012) 31*, 4, 71:1–71:8.

HUANG, H., YIN, K., ZHAO, L., QI, Y., YU, Y., AND TONG, X. 2012. Detail-preserving controllable deformation from sparse examples. *IEEE Trans. on Visualization and Computer Graphics 18*, 8, 1215–1227.

JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. on Graphics (SIGGRAPH 2011) 30*, 4, 78:1–78:8.

JAMES, D. L., AND PAI, D. K. 2002. DyRT: Dynamic Response Textures for Real Time Deformation Simulation With Graphics Hardware. *ACM Trans. on Graphics (SIGGRAPH 2002) 21*, 3, 582–585.

KIM, T., AND JAMES, D. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Trans. on Graphics (SIGGRAPH Asia 2009) 28*, 5, 123:1–123:9.

KIM, T., AND JAMES, D. L. 2012. Physics-based character skinning using multidomain subspace deformations. *IEEE Trans. on Visualization and Computer Graphics 18*, 8, 1228–1240.

KIM, J., AND POLLARD, N. S. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Trans. on Graphics (TOG) 30*, 5, 121.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *Symp. on Computer Animation (SCA)*, 153–160.

KURIHARA, T., AND MIYATA, N. 2004. Modeling deformable human hands from medical images. In *Symp. on Computer Animation (SCA)*, 355–363.

LEHOUCQ, R., SORENSEN, D., AND YANG, C. 1997. ARPACK Users' Guide: Solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods. Tech. rep., Comp. and Applied Mathematics, Rice Univ.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation. In *Proc. of ACM SIGGRAPH 2000*, 165–172.

LIU, L., YIN, K., WANG, B., AND GUO, B. 2013. Simulation and control of skeleton-driven soft body characters. *ACM Trans. on Graphics (SIGGRAPH Asia 2013) 32*, 6, 215.

MAGNENAT-THALMANN, N., LAPERRIRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proc. of Graphics Interface*.

MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. on Graphics (SIGGRAPH 2011) 30*, 4, 37:1–37:11.

NEUMANN, T., VARANASI, K., WENGER, S., WACKER, M., MAGNOR, M., AND THEOBALT, C. 2013. Sparse localized deformation components. *ACM Trans. on Graphics (SIGGRAPH Asia 2013) 32*, 6, 179:1–179:10.

PARDISO, 2015. Parallel Direct Sparse Solver Interface, Pardiso project, http://www.pardiso-project.org and Intel MKL, http://software.intel.com/en-us/articles/intel-mkl.

PONS-MOLL, G., ROMERO, J., MAHMOOD, N., AND BLACK, M. J. 2015. Dyna: A model of dynamic human shape in motion. *ACM Trans. on Graphics (SIGGRAPH 2015) 34*, 4, 120:1–120:14.

TENG, Y., OTADUY, M. A., AND KIM, T. 2014. Simulating articulated subspace self-contact. *ACM Trans. on Graphics (SIGGRAPH 2014) 33*, 4, 106:1–106:9.

TENG, Y., MEYER, M., DEROSE, T., AND KIM, T. 2015. Subspace condensation: Full space adaptivity for subspace deformations. *ACM Trans. on Graphics (SIGGRAPH 2015) 34*, 4, 76:1–76:9.

VON TYCOWICZ, C., SCHULZ, C., SEIDEL, H.-P., AND HILDEBRANDT, K. 2013. An efficient construction of reduced deformable objects. *ACM Trans. on Graphics (SIGGRAPH Asia 2013) 32*, 6, 213.

WANG, Y., JACOBSON, A., BARBIČ, J., AND KAVAN, L. 2015. Linear subspace design for real-time shape deformation. *ACM Trans. on Graphics (SIGGRAPH 2015) 34*, 4, 57:1–57:11.

XU, H., AND BARBIČ, J. 2014. Signed distance fields for polygon soup meshes. In *Proc. of Graphics Interface*, 35–41.

XU, H., LI, Y., CHEN, Y., AND BARBIČ, J. 2015. Interactive material design using model reduction. *ACM Trans. on Graphics (TOG) 34*, 2, 18.

XU, H., SIN, F., ZHU, Y., AND BARBIČ, J. 2015. Nonlinear material design using principal stretches. *ACM Trans. on Graphics (SIGGRAPH 2015) 34*, 4, 75:1–75:11.

YANG, Y., XU, W., GUO, X., ZHOU, K., AND GUO, B. 2013. Boundary-aware multidomain subspace deformation. *IEEE Trans. on Visualization and Computer Graphics 19*, 10, 1633–1645.