Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011)
A. Bargteil and M. van de Panne (Editors)

Physics-based Character Skinning using Multi-Domain Subspace Deformations

Theodore Kim[†1,2] and Doug L. James[3]
University of California, Santa Barbara, Media Arts and Technology Program[1]
University of Saskatchewan, Department of Computer Science[2]
Cornell University, Department of Computer Science[3]

Abstract

We propose a domain-decomposition method to simulate articulated deformable characters entirely within a subspace framework. The method supports quasistatic and dynamic deformations, nonlinear kinematics and materials, and can achieve interactive time-stepping rates. To avoid artificial rigidity, or "locking," associated with coupling low-rank domain models together with hard constraints, we employ penalty-based coupling forces. The multi-domain subspace integrator can simulate deformations efficiently, and exploits efficient subspace-only evaluation of constraint forces between rotated domains using a novel Fast Sandwich Transform *(FST). Examples are presented for articulated characters with quasistatic and dynamic deformations, and interactive performance with hundreds of fully coupled modes. Using our method, we have observed speedups of between three and four orders of magnitude over full-rank, unreduced simulations.*

Categories and Subject Descriptors (according to ACM CCS): Simulation and Modeling [I.6.8]: Types of Simulation—Animation, Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based modeling Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Virtual Reality

1. Introduction

Full-body physics-based character simulations are increasingly common in virtual environments, and the demand for high-resolution million-element character simulations in animation production is increasing [CJ10]. Various approaches have been used to support notoriously expensive physics-based character models, but satisfying the competing goals of interactive performance and increasingly detailed physical realism remains a challenge.

In this paper, we extend physics-based subspace deformation methods to support quasistatic and dynamic articulated characters. Subspace deformation methods, also known as *dimensional model reduction* or *reduced-order deformation* methods, can accelerate the simulation of deformable bodies with arbitrary geometries and non-linear material properties by several orders of magnitude. The speedup usually occurs because the deformation of a model with N vertices is restricted to lie in some linear basis of r carefully selected "displacement modes," and the size of the implicit system to solve is reduced from $N \times N$ to $r \times r$. Since r can be selected independent of N, large speedups can be realized. Unfortunately, for character animation, we often want to simulate expressive deformations of characters with dozens of joints, as well as secondary deformations—both quasistatic pose-space deformations as well as dynamic responses. As the number of modes in a subspace model is increased to resolve these degrees of freedom (DOF), we see diminishing returns in terms of speed and quality. While the first few low-order modes may faithfully capture the bulk motion of a model, higher-order modes resolve spatial detail. Subspace deformation algorithms often make the monolithic assumption that all modes have global support and lie in the same frame of reference, so adding new modes adds to the overall $O(r^2)$ cost (or worse), which complicates treatment of more complex deformable objects, such as articulated characters. As more modes are added, it becomes clear that paying a global cost for increasingly localized behavior in a single frame of reference is not an appealing tradeoff. This limitation significantly complicates the use of subspace methods in applications such as character animation, where a character's limbs may exhibit locally low-rank tissue deformation and large rigid motions.

† e-mail: kim@mat.ucsb.edu

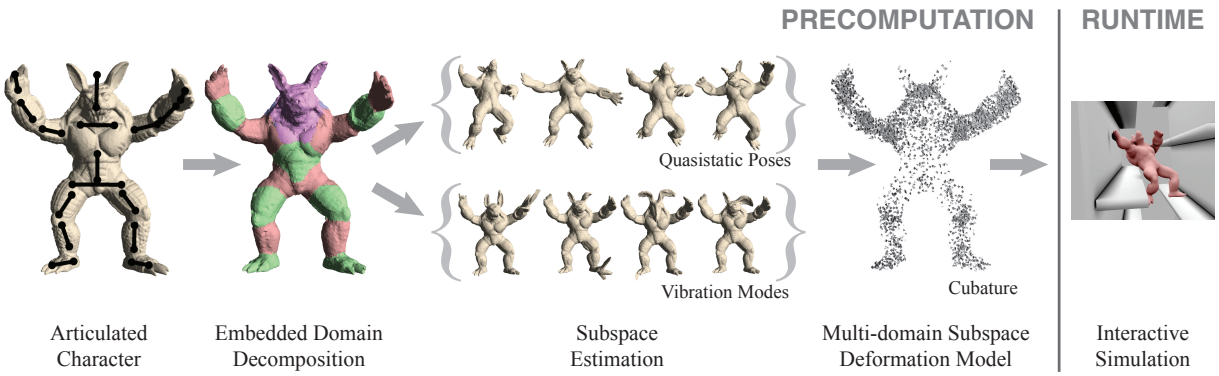


Figure 1: ***Pipeline Overview:*** *Given an articulated character, we generate a discrete tetrahedral mesh and decompose it into bone-associated domains as shown. Each domain's local-frame deformation subspace is estimated by performing weighted PCA on a set of deformations obtained from (1) quasistatic deformations computed for a range of character poses, and (2) linear and derivative modal analysis. Cubature optimization is performed to estimate a reduced-order deformation force for each domain. The resulting domains are held together by inter-domain coupling forces, whose subspace forces are evaluated efficiently using Fast Sandwich Transforms (FSTs) to enable interactive simulation.*

To address these limitations, we propose a novel nonlinear multi-domain subspace deformation method with N-independent runtime simulation cost. We partition the simulation mesh into multiple domains, e.g., the limbs, head, and torso of a character, and attach each to a specific rigid "bone" frame of reference. For each domain, we estimate a subspace deformation model, which has "modes" sufficient to realize both quasistatic deformations needed to resolve example poses, and those needed for dynamic deformations. Since each deformation mode is restricted to a single domain, the domain can be expressive while maintaining a modest rank, and the problem of having to compute global quantities to capture high-DOF articulated behavior is avoided. Each domain's reduced-order force model is computed using existing cubature schemes for nonlinear materials [AKJ08].

A major challenge for multi-domain subspace deformation is to avoiding seam artifacts, e.g., cracks, while coupling domains efficiently. Imposing constraints on all inter-domain vertex pairs, e.g., using Lagrange multipliers, can easily over-constrain the reduced-order models, so we instead use penalty methods which do not explicitly remove DOFs. We use a flexible multibody formulation of domain decomposition where each domain's associated rigid translation and rotation are specified kinematically by the character's pose, and then solve for the modal deformation response.

Unfortunately, computing inter-domain coupling forces involves significant N-dependent costs, as inter-domain vertex positions and penalty (spring) forces must be explicitly computed and resolved in the subspace. Fortunately, we show that we can compute each domain's coupling forces and Jacobians in only $O(r^2)$ time by introducing a precomputation-based *Fast Sandwich Transform* (FST), a simple but useful technique for exploiting the inherent linear dependence of articulated inter-domain computations on time-varying rotation matrix (and related) quantities.

We are able to simulate reduced-order versions of quarter-million-element character models at interactive rates. Hundreds of coupled pose/vibration modes can be efficiently resolved by the multi-domain subspace simulator using FST-based inter-domain constraint resolution. In practice, we observe thousand-fold speedups over the unreduced simulator. An overview our of method, and its associated preprocess and runtime, is shown in Figure 1.

2. Related Work

Various subspace deformation schemes are widely used in character animation, the most widely used being a geometric deformer by the name of *skeletal subspace deformation* (SSD) (or linear blend skinning (LBS), linear enveloping, etc.), and various extensions have been proposed to overcome its notorious shortcomings, e.g., of collapsing elbows [LCF00, KCŽO08]. Most relevant are example-based skinning methods which can support fast animation of physics and artist generated characters, primarily for quasistatic skins (lacking internal dynamics) and dependent just on articulation and possibly muscle activation. *Pose space deformation* techniques [LCF00, SIC01] allow simulated or sculpted character deformations to be pre-generated for key poses (and muscle activations), preprocessed for efficient evaluation (possibly in graphics hardware [KJP02]), then interpolated in a potentially high-dimensional space. Such techniques are now widely used [WPP07, KV08, CJ10], and have inspired various skinning techniques [MG03, JT05]. In contrast to these quasistatic mesh deformers, we use a physics-based subspace model which avoids pose-space interpolation, and generalizes to simulate dynamic and novel

runtime responses; in the absence of dynamics, our method can be seen as an interpolation-free pose-space deformation scheme.

Our physics-based character skinning method builds on prior work in dimensional model reduction [KLM01], also known as reduced-order modeling, or subspace deformation methods. These are best popularized by traditional linear modal analysis methods [Sha90], and have seen extensive use [Sta97, JP02b, HSO03]. To generalize linear modal analysis beyond linear materials and kinematics, subsequent work has shown how to support large deformations [BJ05], as well as nonlinear constitutive materials using cubature schemes [AKJ08]. Although explicitly integrated cubature models can support several hundred modes [CAJ09], implicit (or quasistatic) models have $O(r^3)$ (or worse [BJ05]) complexity, and also struggle to accommodate articulated kinematics. Modal warping [CK05] introduces articulation into linear modal vibration models and is very fast, but the underlying dynamics are still linear oscillators. In contrast, we support nonlinear materials, and accurately model material articulation and deformation using articulated multi-domain subspace deformations. Our approach is based on the multibody formulation of deformable objects [Sha05], which have a long history in computer graphics for the animation of dynamic and coupled bodies [TPBF87, TW88]. Several previous works have employed floating frame formulations [HSO03, BJ05, KSJP08], but only for a single domain. In [KJ09], online model reduction is used to estimate an EigenSkin-style deformation subspace in the absence of collisions. In contrast to these works, we address the problem of stably coupling nonlinear, articulated, reduced-order domain models for character animation, and use multi-domain deformation subspaces derived from both pose-space and modal-analysis reductions.

Various physics-based dynamic deformation schemes have been used for character animation, beyond traditional engineering methods. Multi-resolution simulation of deformable characters was considered in [CGC*02], with bone-attached domains linearized in bone frames, and connected with low-resolution inter-domain vertex constraints; in contrast, we use a full-resolution discretization to derive a fully nonlinear subspace deformation model with domains coupled efficiently using fast sandwich transforms. James and Pai [JP02a] coupled articulated substructures based on precomputed Green's functions [JP99], but resolved inter-domain constraints and rotation sandwiches at the vertex level. Skinned characters have been dynamically augmented in several works. For example, precomputed linear modal models of decoupled torso domains enable GPU simulation of jiggling skinned characters driven by rigidbody bone accelerations [JP02b]. In contrast, we address coupled nonlinear multi-domain deformations, with both pose and mode subspaces. Shape matching methods can also simulate coupled domains [MHTG05], but are unsuitable for animating dynamic character deformations unless very many shape-matching degrees of freedom are simulated [RJ07]. Dynamic skinned deformation models have been simulated using frame-based degrees of freedom with unreduced force evaluation [GBFP11], whereas our frames are specified piecewise with additional pose- and mode-subspaces used to enrich reduced-order deformation physics.

The engineering literature on coupled multibody simulation and substructuring is considerable (for an excellent survey see [WN03]). Various strategies exist for imposing inter-domain coupling constraints, and many established full-rank (i.e. N-dependent) integration methods exist that address this problem. For example, popular methods such as Finite Element Tearing and Interconnect (FETI) and its variants [FLL*01], and Balancing Domain Decomposition by Constraints (BDDC) [Doh03], utilize a large number of hard constraints to ensure that the domain-interface vertices align. Unfortunately they cannot be applied to the case of subspace deformation, because each Lagrange multiplier associated with each hard constraint removes a DOF from the simulation, and subspace methods derive their speed from the fact they have a very limited set of reduced DOFs. A subspace simulation would either lock (i.e., cease to deform adequately), or become over-constrained and produce singular Jacobians. Other substructuring methods use small-deformation linear subspaces that limit the range of applications. In contrast, we impose inter-domain constraints using penalty forces, and rely on the Fast Sandwich Transform (FST) to enable efficient large-deformation subspace integration with articulated domains. Note that the need for FST does not arise in other non-articulated domain decomposition applications, such as multi-domain subspace fluids operate on subspace velocity fields in an Eulerian setting without sandwiched dynamic inter-domain rotations [WST09].

In the unreduced simulation setting, quasistatic deformation models are commonly used when dynamic effects are of secondary importance [TSIF05]. Reduced-order domain decomposition has similarities to higher-order finite element methods, and inter-domain/element penalty forces are often used to enforce continuity, e.g., for Discontinuous Galerkin FEM (DG-FEM) [KMBG08], however, in contrast, our multi-domain subspaces can produce arbitrarily small inter-domain seam errors with increasing subspace rank (see Figure 3). Huang et al. [HLB*06] proposed an unreduced domain decomposition method where at most 6 domains are considered, only linear domain materials are possible, and interface constraints are handled using Lagrange multipliers. While some speedup is observed over [MG04] due to cheaper matrix solves, the integration method still has $\Omega(N)$ cost and is interactive only for a couple thousand tetrahedra.

In concurrent work, [BZ11] proposes a reduced-order domain-decomposition method. It addresses the special case of passive models that can be partitioned into tree-structured reduced-order domains that are connected by interfaces which are small and/or have negligible near-rigid defor-

mations. Domains are preprocessed individually, and domain articulation is determined using a tree-structured recursive approach. In contrast, we optimize our formulation for secondary deformations of active characters, and specify domain articulation via character pose. Our domain-decomposition method supports arbitrarily shaped domains, arbitrary domain connectivity, and inter-domain interfaces can be both large and have significant deformations. Inter-domain continuity constraints are imposed efficiently using soft constraints and the FST, and domains are preprocessed together to estimate nontrivial subspaces for both modal dynamics and pose-space deformations.

3. Multi-Domain Subspace Deformation

We simulate multi-domain systems comprised of reduced-order deformable bodies. The single-domain problem is introduced first, then extended to multiple coupled domains. Computational bottlenecks associated with inter-domain coupling forces will be described in preparation for the following section (§4) on Fast Sandwich Transforms.

3.1. Single-Domain Formulation

We first describe the traditional case of a single body/domain simulated using subspace deformations.

Inertial-frame case: For a deformable body with N vertices at undeformed positions $\mathbf{p} \in \Re^{3N}$, we denote the vertex displacements using the vector $\mathbf{u} \in \Re^{3N}$, and the rank-r reduced displacements as $\mathbf{u} = \mathbf{U}\mathbf{q}$, where $\mathbf{U} \in \Re^{3N \times r}$ is some reduced displacement basis (discussed later), and $\mathbf{q} \in \Re^r$ are the reduced displacement coordinates. We can assume that the domain is somehow fixed in this frame of reference, e.g., to a bone. For compactness, we also introduce the homogeneous coordinate notations $\underline{\mathbf{q}} = \binom{\mathbf{q}}{1} \in \Re^{r+1}$ and $\underline{\mathbf{U}} = [\mathbf{U} \,|\, \mathbf{p}]$, so that the deformed positions $\bar{\mathbf{x}} = \mathbf{p} + \mathbf{u}$ are simply $\bar{\mathbf{x}} = \underline{\mathbf{U}}\underline{\mathbf{q}}$. It follows that the reduced-order Euler-Lagrange equations of motion for the deformable body can be written as [BJ05, AKJ08]

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{f}_{int}(\mathbf{q}) = \mathbf{f}_{ext},$$

where $\mathbf{M} \in \Re^{r \times r}$ and $\mathbf{C} \in \Re^{r \times r}$ are respectively the reduced mass and damping matrices, $\mathbf{f}_{ext}$ are external forces projected into the subspace, and the overdots denote differentiation with respect to time; the $\mathbf{f}_{int}(\mathbf{q})$ term is the internal force response, and can correspond to any arbitrary nonlinear constitutive model.

Moving-frame case: Consider a multibody formulation where each deformable domain also has an associated rigid translation $\mathbf{t} \in \Re^3$ and rotation $\mathbf{R} \in \Re^{3\times 3}$. The k^{th} vertex's position in its domain's local frame is

$$\bar{\mathbf{x}}_k = \mathbf{p}_k + \mathbf{U}_k\mathbf{q} = \underline{\mathbf{U}}_k\underline{\mathbf{q}}, \tag{1}$$

with $\mathbf{U}_k$ the 3 rows of $\mathbf{U}$ corresponding to the k^{th} vertex. Its world-frame position is

$$\mathbf{x}_k = \mathbf{R}\bar{\mathbf{x}}_k + \mathbf{t} = \mathbf{R}(\mathbf{p}_k + \mathbf{U}_k\mathbf{q}) + \mathbf{t} = \mathbf{R}\underline{\mathbf{U}}_k\underline{\mathbf{q}} + \mathbf{t}.$$

In order to incorporate the dynamics of these rigid components into our equations, we use the generalized Newton-Euler equations [MT92, Sha05],

$$\begin{bmatrix} \mathbf{M_{tt}} & \mathbf{R}\widetilde{\mathbf{m}}^T_{\mathbf{t}\omega} & \mathbf{RM_{tq}} \\ & \mathbf{M}_{\omega\omega} & \mathbf{M_{\omega q}} \\ \text{sym.} & & \mathbf{M} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{t}} \\ \dot{\omega} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{ext,\mathbf{t}} \\ \mathbf{f}_{ext,\omega} \\ \mathbf{f}_{ext} - \mathbf{C}\dot{\mathbf{q}} - \mathbf{f}_{int} \end{bmatrix} + \begin{bmatrix} \mathbf{f}_{qv,\mathbf{t}} \\ \mathbf{f}_{qv,\omega} \\ \mathbf{f}_{qv} \end{bmatrix}$$

where $\dot{\omega}$ denotes angular acceleration, $\mathbf{f}_{ext,\mathbf{t}}$ and $\mathbf{f}_{ext,\omega}$ denote the translational and angular components of the external force, and $\begin{bmatrix} \mathbf{f}_{qv,\mathbf{t}} & \mathbf{f}_{qv,\omega} & \mathbf{f}_{qv} \end{bmatrix}^T$ denotes the quadratic velocity vector of centrifugal and Coriolis forces; the left hand side block matrix is the time-varying mass matrix.

For character skinning, let each domain's rigidbody motion be specified, e.g., via an attached bone, so we only need to solve for the deformation coordinates, $\mathbf{q}$, via

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{f}_{int}(\mathbf{q}) = \mathbf{f}_{ext} + \mathbf{f}_{qv} + \mathbf{f}_{rigid} \tag{2}$$

where the rigid-frame coupling forces are

$$\mathbf{f}_{rigid} = -\left(\mathbf{M}^T_{\mathbf{tq}}\mathbf{R}^T\ddot{\mathbf{t}} + \mathbf{M}^T_{\omega\mathbf{q}}\dot{\omega}\right) \tag{3}$$

which convert prescribed rigid-body (e.g., bone) accelerations into subspace deformation forces [JP02b].

Efficient evaluation: We use the cubature approach of [AKJ08] to approximate the $\mathbf{f}_{int}(\mathbf{q})$ internal deformation force (and Jacobian) in $O(r^2)$ (and $O(r^3)$) time per domain. As in [KJ09], cubature schemes are evaluated using invertible elements [ITF04, TSIF05] to avoid inversion-related indefiniteness problems that easily arise in character animation. We refer the reader to Appendix A for details on evaluating the quadratic velocity force, mass matrix and $\mathbf{f}_{rigid}$.

3.2. Multi-Domain Formulation

In the multi-domain setting, the equations of motion for the i^{th} domain are similar,

$$\mathbf{M}^i\ddot{\mathbf{q}}^i + \mathbf{C}^i\dot{\mathbf{q}}^i + \mathbf{f}_{int}(\mathbf{q}^i) = \mathbf{f}^i_{ext} + \mathbf{f}^i_{qv} + \mathbf{f}^i_{rigid} + \mathbf{f}^i_c \tag{4}$$

but now include interactions with neighboring domains, $j \in \mathcal{N}_i$, via the coupling force,

$$\mathbf{f}^i_c = \sum_{j \in \mathcal{N}_i} \mathbf{f}^{ij}_c \tag{5}$$

which enforces continuity at interface nodes (see Figure 2).

Figure 2: ***Inter-domain penalty constraints*** *are used to enforce continuity constraints and avoid seams and cracking artifacts. Spring-damper penalty forces are used to avoid over-constraining the subspace dynamics via explicit constraints, which could either be infeasible or cause locking to occur. Efficiently evaluating subspace coupling forces between rotated domains is achieved using Fast Sandwich Transforms (§4).*

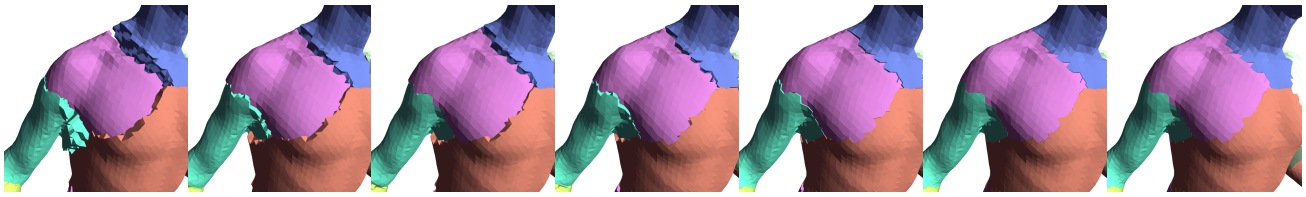


Figure 3: ***Inter-domain constraint satisfaction vs domain rank*** *reveals rapid gap-error reduction with increasing domain rank on the chest and shoulder of a human body mesh. The basis was trained on deformations computed from motion capture data.*

3.2.1. Inter-Domain Coupling Forces

Spring coupling forces: In order to avoid locking in the subspace models, we apply penalty-force constraints that do not explicitly remove degrees of freedom. We use linear springs at interface vertices, $k = 1 \ldots n$, to produce unreduced external forces of the form

$$\mathbf{F}_k^{ij} = -\kappa a_k \left(\mathbf{x}_k^i - \mathbf{x}_k^j\right) \tag{6}$$

$$= -\kappa a_k \left((\mathbf{R}^i \underline{\mathbf{U}}_k^{ij} \underline{\mathbf{q}}^i + \mathbf{t}^i) - (\mathbf{R}^j \underline{\mathbf{U}}_k^{ji} \underline{\mathbf{q}}^j + \mathbf{t}^j)\right), \tag{7}$$

where κ is the spring constant, and a_k is the effective interfacial area for vertex k. Here $\mathbf{U}^i$ denotes the displacement basis for domain i, and $\mathbf{U}^{ij}$ is a submatrix of rows in $\mathbf{U}^i$ corresponding to vertices along the i, j interface. For simplicity, we will refer to each domain's rank as r without loss of generality, although in practice each domain can have different rank. The vector of all n spring forces is then

$$\mathbf{F}^{ij} = -\kappa \mathbf{A} \left(\mathbf{R}^i \underline{\mathbf{U}}^{ij} \underline{\mathbf{q}}^i - \mathbf{R}^j \underline{\mathbf{U}}^{ji} \underline{\mathbf{q}}^j + \mathbf{I}_n (\mathbf{t}^i - \mathbf{t}^j)\right), \tag{8}$$

where $\mathbf{A} = \mathsf{diag}_n(a_k \mathbf{I}_3)$ is a diagonal area-weighting matrix; $\mathbf{I}_n \in \Re^{3n \times 3}$ is a column vector of n 3-by-3 identity matrices. For notational clarity we assume that $\mathbf{R} \in \mathbb{R}^{3\times 3}$ can also represent its block-diagonalized version, $\mathsf{diag}_n(\mathbf{R}) \in \mathbb{R}^{3n\times 3n}$ as needed without introducing extra "diag" notation. For example, the rotated 3-vector entries $\mathbf{u}_k$ of the displacement vector, $\mathbf{u} \in \mathbb{R}^{3n}$, can be written simply as $\mathbf{Ru}$ instead of $\mathsf{diag}_n(\mathbf{R})\mathbf{u}$; similarly $\mathbf{R}\underline{\mathbf{U}} \equiv \mathsf{diag}_n(\mathbf{R})\underline{\mathbf{U}}$.

Subspace coupling forces: Each interface spring contributes to the domain interface's coupling force vector, and has the following mathematical structure

$$\mathbf{f}_c^{ij} = \sum_{k=1}^{n} (\mathbf{U}_k^{ij})^T (\mathbf{R}^i)^T \mathbf{F}_k^{ij} \tag{9}$$

$$= (\mathbf{R}^i \mathbf{U}^{ij})^T \mathbf{F}^{ij} = (\mathbf{U}^{ij})^T (\mathbf{R}^i)^T \mathbf{F}^{ij} \tag{10}$$

$$= -\kappa \Big([(\mathbf{U}^{ij})^T \mathbf{A} \underline{\mathbf{U}}^{ij}] \underline{\mathbf{q}}^i - [(\mathbf{U}^{ij})^T \mathbf{A} (\mathbf{R}^i)^T \mathbf{R}^j \underline{\mathbf{U}}^{ji}] \underline{\mathbf{q}}^j + [(\bar{\mathbf{U}}^{ij})^T \mathbf{A} \mathbf{I}_n] (\mathbf{R}^i)^T (\mathbf{t}^i - \mathbf{t}^j)\Big) \tag{11}$$

These n-vertex summations can be a force (and Jacobian) evaluation bottleneck for detailed models. While the summation can be precomputed for some terms, e.g., $[(\mathbf{U}^{ij})^T \mathbf{A} \underline{\mathbf{U}}^{ij}]$, the colored term involves a time-varying quanty (in red) "sandwiched" between other (blue) matrices which precludes precomputation. We address their efficient evaluation using Fast Sandwich Transforms in §4.

Damped coupling forces: In order to avoid undesirable inter-domain jiggling artifacts due to stiff inter-domain springs, we can optionally damp inter-domain spring forces as follows. Specifically, consider the proportional damping force on vertex k,

$$\mathbf{d}_k^{ij} = -\kappa_d a_k \left(\dot{\mathbf{x}}_k^i - \dot{\mathbf{x}}_k^j\right) \tag{12}$$

$$= -\kappa_d a_k \Big(\dot{\mathbf{R}}^i \underline{\mathbf{U}}_k^{ij} \underline{\mathbf{q}}^i + \mathbf{R}^i \mathbf{U}_k^{ij} \dot{\mathbf{q}}^i + \dot{\mathbf{t}}^i \tag{13}$$

$$- \dot{\mathbf{R}}^j \underline{\mathbf{U}}_k^{ji} \underline{\mathbf{q}}^j - \mathbf{R}^j \mathbf{U}_k^{ji} \dot{\mathbf{q}}^j - \dot{\mathbf{t}}^j\Big) \tag{14}$$

which leads to a subspace damping force

$$\mathbf{f}_{c,damp}^{ij} = (\mathbf{R}^i \mathbf{U}^{ij})^T \mathbf{d}^{ij} \tag{15}$$

$$= -\kappa_d \Big([(\mathbf{U}^{ij})^T \mathbf{A} (\mathbf{R}^i)^T \dot{\mathbf{R}}^i \underline{\mathbf{U}}^{ij}] \underline{\mathbf{q}}^i - [(\mathbf{U}^{ij})^T \mathbf{A} (\mathbf{R}^i)^T \dot{\mathbf{R}}^j \underline{\mathbf{U}}^{ji}] \underline{\mathbf{q}}^j + [(\mathbf{U}^{ij})^T \mathbf{A} \mathbf{U}^{ij}] \dot{\mathbf{q}}^i - [(\mathbf{U}^{ij})^T \mathbf{A} (\mathbf{R}^i)^T \mathbf{R}^j \mathbf{U}^{ji}] \dot{\mathbf{q}}^j + [(\mathbf{U}^{ij})^T \mathbf{A} \mathbf{I}_n] (\mathbf{R}^i)^T (\dot{\mathbf{t}}^i - \dot{\mathbf{t}}^j)\Big). \tag{16}$$

For sufficiently small $\dot{\mathbf{R}}$ and $\dot{\mathbf{t}}$ motions, we can approximate the damping force as

$$\mathbf{f}_{c,damp}^{ij} \approx -\kappa_d \left[[(\mathbf{U}^{ij})^T \mathbf{A} \mathbf{U}^{ij}] \dot{\mathbf{q}}^i - [(\mathbf{U}^{ij})^T \mathbf{A} (\mathbf{R}^i)^T \mathbf{R}^j \mathbf{U}^{ji}] \dot{\mathbf{q}}^j\right]. \tag{17}$$

The damping force Jacobian (w.r.t $\dot{\mathbf{q}}$) is straightforward. Again, Fast Sandwich Transforms are needed to evaluate interface damping forces without n-dependent summations.

4. Fast Sandwich Transforms

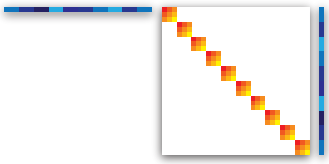
A key challenge for multi-domain coupling is to enable efficient inter-domain subspace communication despite rotated domains. This section shows how to avoid all n-dependent runtime computations when computing inter-domain coupling forces, and thus avoid considering many thousands of vertex computations. The basic observation is that the seemingly n-dependent computations are actually linear in the entries of the rotation matrices, which only contain a small, constant number of terms. By using a carefully structured preprocess, only r-dependent computations become needed at runtime. This explicit preprocessing avoids n-dependent codes that are commonly generated, e.g., by computer algebra systems for automatic force code generation.

4.1. Rotation sandwiches and other slow fare

Many computations involve multiplication between constant matrices, like $\underline{\mathbf{U}}$, which are known ahead of time and can be pre-multiplied during setup. For example, consider a dot-product between two $3n$ displacement vectors, $\mathbf{u}$ and $\mathbf{v}$, which are constant and known before the simulation starts. Clearly computing $\mathbf{u}^T\mathbf{v} = \sum_{k=1}^{n} \mathbf{u}_k^T \mathbf{v}_k$ is n-dependent, but can be trivially preprocessed away. Similarly, the first term of (11) has the constant sub-matrix $(\mathbf{U}^{ij})^T\underline{\mathbf{U}}^{ij} \in \Re^{r\times(r+1)}$, where the two matrices can be premultiplied once at $O(r^2n)$ cost to avoid n-dependent runtime costs.

One tiny sandwich: The situation is a little more complicated if a time-dependent quantity, like a rotation, is "sandwiched" in between $\mathbf{u}$ and $\mathbf{v}$:

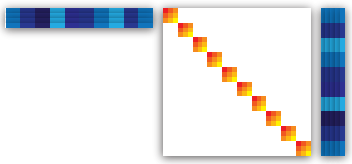
$$\mathbf{u}^T\mathbf{R}\mathbf{v} = \tag{18}$$



Here bluish entries are constant, whereas the reddish rotation blocks are not. Unfortunately, the time-dependent rotation is sandwiched between the constant matrices, which complicates premultiplication, and suggests that $O(n)$ work is needed at runtime.

Bigger sandwiches: The situation is far worse for many terms we will meet later – time-varying rotations and the like are sandwiched between vectors, matrices, and higher-order tensors. For example, the second term in (11) has a nonconstant $r\times(r+1)$ matrix with a relative rotation (call it $\mathbf{R}$) sandwiched inside,

$$(\mathbf{U}^{ij})^T\mathbf{R}\underline{\mathbf{U}}^{ij} = \tag{19}$$



leading to an $O(r^2n)$ runtime evaluation cost.

4.2. The Fast Sandwich Transform

The basic idea of the Fast Sandwich Transform (FST), is to preprocess away n-dependent computations by exploiting the linearity of time-varying rotation parameters. The FST is essentially a preprocess-based fast summation technique. For example, the simple dot-product computation, $\mathbf{u}^T\mathbf{R}\mathbf{v}$, is linear in the only time-varying parameters: the 9 entries of $\mathbf{R}$. To exploit this linearity, we can introduce the 3x3 basis matrices

$$\mathbf{E}^{\mu\nu} = \begin{bmatrix} \delta_{\mu1}\delta_{\nu1} & \delta_{\mu1}\delta_{\nu2} & \delta_{\mu1}\delta_{\nu3} \\ \delta_{\mu2}\delta_{\nu1} & \delta_{\mu2}\delta_{\nu2} & \delta_{\mu2}\delta_{\nu3} \\ \delta_{\mu3}\delta_{\nu1} & \delta_{\mu3}\delta_{\nu2} & \delta_{\mu3}\delta_{\nu3} \end{bmatrix}, \quad \begin{matrix} \mu = 1\ldots3 \\ \nu = 1\ldots3 \end{matrix} \tag{20}$$

so that, if the elements of $\mathbf{R}$ are $R_{\mu\nu}$, then we can write

$$\mathbf{R} = \mathbf{E}^{\mu\nu}R_{\mu\nu}, \qquad \text{(implied sum over } \mu, \nu) \tag{21}$$

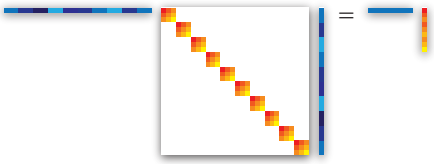
and thus

$$\mathbf{u}^T\mathbf{R}\mathbf{v} = \mathbf{u}^T\mathbf{E}^{\mu\nu}R_{\mu\nu}\mathbf{v} = \left(\mathbf{u}^T\mathbf{E}^{\mu\nu}\mathbf{v}\right)R_{\mu\nu} \tag{22}$$

$$= A^{\mu\nu}R_{\mu\nu}, \tag{23}$$

which resembles a dot product. Consequently we can restructure the computation during preprocessing to obtain

$\mathbf{u}^T\mathbf{R}\mathbf{v} =$



The generalization to sandwiches with constant matrix (or tensor) "bread" is straightforward,

$$\mathbf{U}^T\mathbf{R}\underline{\mathbf{U}} = \mathbf{U}^T\mathbf{E}^{\mu\nu}R_{\mu\nu}\underline{\mathbf{U}} = \left(\mathbf{U}^T\mathbf{E}^{\mu\nu}\underline{\mathbf{U}}\right)R_{\mu\nu} \tag{24}$$

$$= \mathbf{A}^{\mu\nu}R_{\mu\nu}, \tag{25}$$

and is now just a linear superposition of 9 r-by-$(r+1)$ precomputed matrices, $\mathbf{A}^{\mu\nu}$. In this way, we can exploit linearity to preprocess away sandwiched rotations and similar time-dependent matrices, e.g., $[(\mathbf{R}^i)^T\dot{\mathbf{R}}^j]$, for fast n-independent runtime evaluation. We observe that this process can also be thought of in terms of tensors, $\mathbf{A}\otimes R$, where $\mathbf{A} \in \Re^{r\times r\times 9}$, $R \in \Re^9$ and $\otimes$ denotes a mode-3 tensor product.

4.3. Application

Using Fast Sandwich Transforms, all of the afore-mentioned inter-domain coupling spring forces (§3.2.1) can be evaluated in $O(r^2)$ time per interface, thereby avoiding n-dependent vertex force computations at runtime. Specifically, we precompute one FST to evaluate each interface's coupling spring force $\mathbf{f}_c^{ij}$ in (11), and another FST to evaluate the damping force $\mathbf{f}_{c,damp}^{ij}$ in (17). Spring-damper force Jacobians are needed for implicit integration, and can also be evaluated using the same FSTs. In addition, for each domain we can precompute two FSTs to efficiently evaluate the quadratic-velocity force in (26) (see Appendix A).

5. Character Animation Pipeline

The quality of the character deformation and dynamics depends directly on the quality of the subspace basis in each domain. We have constructed a pipeline that automatically constructs a high quality basis given a skeleton embedded in a tetrahedral mesh. An overview is shown in Figure 1.

5.1. Preprocess

The user can provide a character's tetrahedral mesh and skeletal articulation. If no piecewise-rigid rigging is provided, we constrain the tetrahedra intersected by the bones of the skeleton to transform kinematically along with the associated bones, and the mesh is partitioned into body-part domains by associating each tetrahedron with the nearest

bone. Alternately, skinning information from automatic rigging software, e.g., Pinocchio [BP07], can be used to generate a partitioning: we assign each tetrahedron the skinning weight of the nearest surface vertex, and then assign it to the bone with the highest relative skinning weight.

Next, we build a subspace basis for each domain. We randomly sample the joint space of the skeleton, but respect the model's joint limits to prevent extreme interpenetrations. We then perform a quasistatic solve over the entire tetrahedral mesh for each joint configuration. These quasistatic solves can be computed in parallel. We found full dynamics solves, e.g., mocap sequences, to be inadvisable: a combination of large joint accelerations and extreme element inversions tend to cause the Newton solves to diverge. Principal component analysis (PCA) is then performed on the simulation subsets corresponding to each domain, and the most relevant modes are added to each domain basis. If relevant pose spaces are available, e.g., using motion capture data, these skeletal configurations can be used in lieu of the random joint samples.

We introduce dynamics information into the basis by computing a vibration analysis of the entire mesh; we use weighted linear modal analysis and modal derivatives [BJ05]. Per-domain PCA is again performed on the results, and the most relevant modes are added to each domain basis (and orthogonalized for better numerical conditioning). We found that allocating half of the basis to vibration modes and half to quasistatic poses works well in practice. Cubature optimization requires training data that samples the deformation subspace, and we use all projected quasistatic poses plus an equivalent number of eigenvalue-weighted random samples from the vibration analysis [AKJ08].

5.2. Runtime Simulation

At runtime, our simulator reads in a new skeleton pose at each time step and solves for the new deformation. The pose can come from an arbitrary source, such as motion capture data or an articulated rigid body solver. Once the simulator receives the new pose, it transforms any constrained cubature vertices, and performs a Newton solve to determine the new deformation and dynamics. Each Newton iteration constructs a symmetric block-sparse system (Fig. 5) that is then solved. The systems we encountered were sufficiently small that a direct Cholesky solver was used on the resulting matrix, but for larger systems, conjugate gradient could be employed instead. Collisions are detected using Bounded Deformation Trees [JP04]. Our implementation uses an IMEX integrator wherein $\mathbf{f}_{qv}$ and $\mathbf{f}_{rigid}$ forces are integrated explicitly, but subspace dynamics, including $\mathbf{f}_c$ and $\mathbf{f}_{c,damp}$, are integrated implicitly, e.g., using implicit subspace Newmark [BJ05]. The rigid velocities and accelerations needed to compute $\mathbf{f}_{qv}$ and $\mathbf{f}_{rigid}$ were obtained by plugging the translations and rotations into the same update equations being used by the subspace solver, e.g., also implicit Newmark.

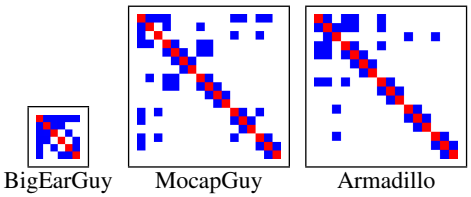


Figure 5: ***Block-sparse matrix structure:*** *Each block row corresponds to one domain, and blue off-diagonal entries correspond to coupling force Jacobians. For parallelization, we sent each block row to a separate core.*

6. Results

Examples statistics are reported in Table 1, whereas performance statistics are given in Table 2. Examples are demonstrated in Figure 4, and our accompanying video. Timings were conducted on an 8-core Apple Mac Pro with 2.26 Ghz Intel Xeon cores (Nehalem, E5520) and 32 GB of RAM. Full-rank Newton solves for all training examples were run to two digits of precision. MocapGuy was limited to 15 Newton iterations per frame, whereas BigEarGuy and Armadillo were limited to 75 iterations since their randomized poses lacked frame-to-frame coherence, and each solution had to be found essentially from scratch. Temporally coherent poses could reduce the running time of the full-rank solve (e.g. for MocapGuy), and potentially the overall speedups in Table 2. We maintain that even with such speedups, our method remains orders of magnitude faster.

Reduced order implicit and quasistatic solves were limited to at most 3 Newton iterations per frame. The timestep was set to $\Delta t = 1/120s$ in all cases. For the BigEarGuy and Armadillo examples, we used 100 joint samples, 160 linear modal analysis modes, and 100 modal derivatives. The examples both set $\kappa = 200$ and respectively $\kappa_d = 0.1, \kappa_d = 0.01$. For the MocapGuy example, we took advantage of the large amount of available data and used 3600 motion capture frames, and set $\kappa = 5000, \kappa_d = 0.1$. In all of the examples, we observed that our penalty-based inter-domain constraint approach was able to sufficiently resolve inter-domain contacts so that cracks, or other artifacts, were not apparent (see Figure 3). In the BigEarGuy and Armadillo examples (see Figure 4), we used the results of an Open Dynamics Engine (ODE) [Smi06] simulation, then computed character deformations for the resulting frame time-series. Consequently, in the Armadillo examples, vertex-level contact deformations are not fully realized. In the MocapGuy example, we used motion-capture data provided with the Pinocchio [BP07] distribution (`http://www.mit.edu/~ibaran/autorig/pinocchio.html`)

OpenMP parallelization was used to compute the cubature and coupling forces (see Table 2), as these stages parallelize naturally. The block diagonal entries are by far the most expensive, as they incorporate the cubature stiffness matrices,

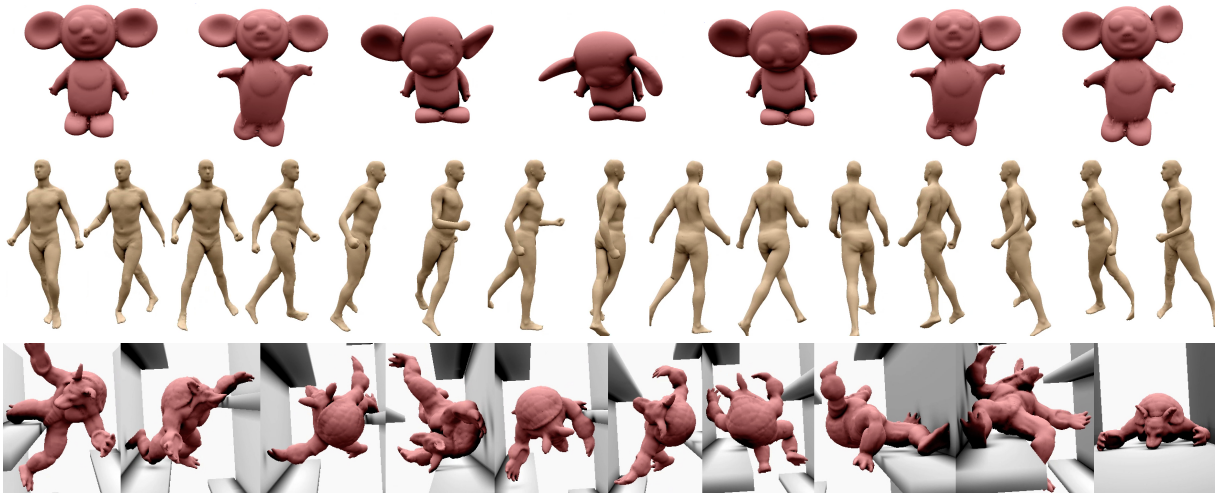


Figure 4: ***Character examples, top to bottom:*** *Images are high resolution; please zoom in for more detail. The* ***BigEarGuy*** *model stress-tests our subspace model by undergoing a "jumping-jack" motion that simultaneously exercises all of the articulated skeleton's joint limits. The* ***MocapGuy*** *model performs a cross-walking motion not seen during the subspace construction and no visible inter-domain cracks form. The* ***Armadillo*** *model efficiently handles collisions and large accelerations while falling through an obstacle course. An additional 'Armadillo pachinko' example can be seen in the supplemental video.*

Model	Nodes	Tets	Domains	Interfaces	Interface Nodes	Joints	QS Solve Time (unreduced)
BigEarGuy	36070	181751	6	10	2683	5	507 sec/frame
MocapGuy	54370	269139	17	26	5005	-	263 sec/frame
Armadillo	39152	194788	16	18	4525	13	415 sec/frame

Table 1: ***Model Statistics:*** *We report the number of tetrahedral elements (Tets) and nodes in the simulation mesh (Nodes), the number of domains (Domains) and domain-domain interfaces (Interfaces), the number of nodes on the interfaces (not double-counted in "Nodes"), and joints for jointed characters. Finally the average time for frame-to-frame quasistatic solves are reported. Dynamics solves would consume more time, since velocities and accelerations would also need to be resolved.*

and an eigensystem must be solved for each cubature point in order to handle element inversion [TSIF05]. We found the hybrid eigensolver from [Kop08] to be faster than LAPACK.

7. Limitations and Future Work

Although inter-domain gaps do not occur in the final model by virtue of seam-vertex averaging, a smoother spatial reconstruction or higher-order interface constraints may be required to guarantee high-order smoothness (c.f. [KMBG08]). Our implementation does not currently detect or resolve self collisions, however contact forces could be integrated by the system provided the basis is enriched, e.g., pose-level self contacts near joints could be accomodated by using precomputed poses with joint-area self-contacts resolved. Domain decomposition could be optimized to produce domains which reduce simulation costs, and deformation basis rank for a given error. Instead of a piecewise-rigid frame formulation, one could use skinned frames (as in EigenSkin), to provide inherently smoother frame blending. However this choice would complicate subspace dynamics and cubature training; nevertheless, similar FST ideas can be applied to that case. Domain decomposition is ideally suited to parallel implementations, however the subspace solver is sufficiently fast in our current problems that there is limited opportunity for parallel speedups. For example, on an 8-core machine we observe a 2x-3x speedup in force/Jacobian evaluations. We expect that larger offline examples with more domains and reduced DOFs will provide more opportunities for parallelism, and challenges for block-sparse linear system solvers. Finally, muscle activations were not considered herein, but could easily be included in the pose-space formulation, with shapes/displacements modeled using a linear superposition of local-frame body forces.

Appendix A: Efficient evaluation of mass matrices and quadratic velocity force

We efficiently evaluate the mass matrix and quadratic velocity vector in $O(r^2)$ time as follows.

The quadratic velocity force is given by

$$\mathbf{f}_{qv} = -[\mathbf{U}^T \mathsf{M} \cdot \tilde{\omega}^2 \cdot \underline{\mathbf{U}}]\underline{\mathbf{q}} - [2\mathbf{U}^T \mathsf{M} \cdot \tilde{\omega} \cdot \mathbf{U}]\dot{\mathbf{q}} \qquad (26)$$

where M is the domain's unreduced mass matrix. Here $\tilde{\omega}$

Model	Rank	Cubature Tets	FST?	Coupling (ms)	Cubature (ms)	Cholesky (ms)	FPS	Total (ms)	Speedup
BigEarGuy	40/240	562	No	167 (85%)	22 (12%)	5.9 (3%)	5.1	196	
	40/240	562	Yes	3.3 (10%)	22 (64%)	7.6 (23%)	29.6	34	14900x
	30/180	404	No	118 (87%)	13 (10%)	2.7 (2%)	7.4	135	
	30/180	404	Yes	2.1 (11%)	13 (66%)	3.7 (19%)	50.1	20	25300x
	20/120	277	No	83 (89%)	8 (9%)	1.2 (1%)	10.7	93	
	20/120	277	Yes	1.7 (15%)	8 (67%)	1.4 (13%)	90.0	11	46000x
MocapGuy	20/340	1368	No	84 (66%)	34 (27%)	8.2 (6.4%)	7.8	128	
	20/340	1368	Yes	2.2 (5%)	34 (75%)	8.3 (18%)	22.2	45	5800x
Armadillo	40/640	3748	No	796 (77%)	127 (12%)	107 (10%)	0.96	1040	
	40/640	3748	Yes	8.3 (3%)	128 (52%)	100 (41%)	4.1	243	1700x
	30/480	1060	No	523 (80%)	3.4 (6%)	48 (8%)	1.6	625	
	30/480	1060	Yes	5.2 (6%)	3.4 (38%)	44 (49%)	11	91	4500x
	20/320	728	No	332 (88%)	20 (5%)	17 (4%)	2.7	370	
	20/320	728	Yes	2.9 (7%)	20 (48%)	14 (33%)	24	42	9900x

Table 2: ***Performance Statistics:*** *For each example, we report the rank of each domain and the total model rank (summed over all domains), the number of tetrahedral elements used in the cubature scheme (total for all domains; 5% training error); and whether or not Fast Sandwich Transforms are enabled (FST?). Timing breakdowns (in milliseconds) are provided, and percentages of total timestep/frame cost. In all cases, without FST the inter-domain coupling force/Jacobian evaluation is a bottleneck, but with FST the bottleneck shifts to cubature force/Jacobian evalution and the block-sparse Cholesky solve. The total frame cost is reported, and compared to the non-subspace quasistatic solver cost (in Table 1), and reveal significant speedups (1700x—46000x) depending on domain rank. See §6 for further details. OpenMP parallelization was used to obtain modest 2x-3x speedups on Cubature and Coupling force/Jacobian evaluation; Cholesky solves are not parallelized.*

is the auto-block-diagonalized matrix with skew-symmetric "$\omega\times$" 3x3 blocks. The quadratic velocity force is integrated explicitly, and can be efficiently evaluated using two fast sandwich transforms (§4), as the colors indicate.

Mass matrix blocks: The modal mass matrix, $\mathbf{M}$ is constant and can be precomputed. However to evaluate the linear-defo and angular-defo coupling forces, we require $\mathbf{M_{tq}}$ and $\mathbf{M_{\omega q}}$, respectively. First, $\mathbf{M_{tq}} = \sum_k m_k \mathbf{U}_k$, where m_k is the mass of the k^{th} node, is constant and can be trivially precomputed. Second, $\mathbf{M_{\omega q}}$ is the time-dependent rotation-deformation tensor, defined as,

$$\mathbf{M_{\omega q}} = \sum_k m_k \tilde{\mathbf{x}}_k \mathbf{U}_k, \tag{27}$$

where $\mathbf{U}_k$ are the rows of $\mathbf{U}$ corresponding to the k^{th} vertex, $\tilde{\mathbf{x}}_k$, and the tilde represents the skew-symmetric cross product. This can be stated as:

$$\mathbf{M_{\omega q}} = \sum_k m_k \left(\widetilde{\mathbf{p}}_k \mathbf{U}_k + (\widetilde{\mathbf{U}}_k \cdot \mathbf{U}_k) \otimes \mathbf{q} \right). \tag{28}$$

The $\widetilde{\mathbf{p}}_k \mathbf{U}_k$ and $\widetilde{\mathbf{U}}_k \mathbf{U}_k$ terms can be precomputed and summed. A mode-3 product with an $\Re^{3\times r\times r}$ tensor must be computed at runtime, which takes $O(r^2)$ time.

Acknowledgements: This work was supported in part by the National Science Foundation (CAREER-0430528, EMT-CompBio-0621999), the National Institutes of Health (NIBIB/NIH R01EB006615), NSERC (Many-core Physically Based Simulations), the Alfred P. Sloan Foundation, and donations from Intel, Pixar, and Autodesk. DLJ acknowledges early discussions with Dinesh Pai on precomputation-based domain decomposition and the rotation-sandwich problem [JP02a], and also collaborations with students under CAREER-0430528: early work with Christopher Twigg on articulated tree-structured reduced-order domains for Krysl-style simulation (unpublished) and data-driven animation of botanical systems [JTCW07], and later with Jernej Barbic on simulating reduced-order StVK models [BJ05] in the context of articulated tree-structured reduced-order domains (that led to [BZ11]). Reduced-order domain decomposition work was supported in part by NIBIB/NIH R01EB006615, and we thank Jaydev Desai for his infinite patience and support. We thank Changxi Zheng, Jeffrey Chadwick, and Steven An for collaboration, and the anonymous reviewers for their helpful feedback. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or others.

References

[AKJ08] AN S. S., KIM T., JAMES D. L.: Optimizing Cubature for Efficient Integration of Subspace Deformations. *ACM Trans. on Graphics 27*, 5 (Dec. 2008), 165.

[BJ05] BARBIČ J., JAMES D. L.: Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. on Graphics 24*, 3 (Aug. 2005), 982–990.

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics 26*, 3 (July 2007), 72:1–72:8.

[BZ11] BARBIČ J., ZHAO Y.: Real-time large-deformation substructuring. *ACM Trans. on Graphics 30* (2011).

[CAJ09] CHADWICK J. N., AN S. S., JAMES D. L.: Harmonic Shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 119:1–119:10.

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive Skeleton-Driven Dynamic Deformations. *ACM Trans. on Graphics 21*, 3 (July 2002), 586–593.

[CJ10] CLUTTERBUCK S., JACOBS J.: A physically based approach to virtual character deformations. SIGGRAPH 2010 Talks, 2010.

[CK05] CHOI M. G., KO H.-S.: Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (Jan./Feb. 2005), 91–101.

[Doh03] DOHRMANN C. R.: A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal of Scientific Computation 25* (January 2003), 246–258.

[FLL*01] FARHAT C., LESOINNE M., LETALLEC P., PIERSON K., RIXEN D.: FETI-DP: a dual-primal unified FETI method - part I: A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering 50* (2001).

[GBFP11] GILLES B., BOUSQUET G., FAURE F., PAI D. K.: Frame-based elastic models. *ACM Trans. Graph. 30* (April 2011), 15:1–15:12.

[HLB*06] HUANG J., LIU X., BAO H., GUO B., SHUM H.: An efficient large deformation method using domain decomposition. *Computers & Graphics 30*, 6 (2006), 927–935.

[HSO03] HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive deformation using modal analysis with constraints. In *Graphics Interface 2003* (June 2003), pp. 247–256.

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *ACM SIGGRAPH Symposium on Computer Animation* (July 2004), pp. 131–140.

[JP99] JAMES D. L., PAI D. K.: ArtDefo: Accurate real time deformable objects. In *Computer Graphics (Proceedings of SIGGRAPH 99)* (Aug. 1999), pp. 65–72.

[JP02a] JAMES D., PAI D.: Real time simulation of multizone elastokinematic models. In *Proceedings of IEEE ICRA'02* (2002), vol. 1, IEEE, pp. 927–932.

[JP02b] JAMES D. L., PAI D. K.: DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. on Graphics 21*, 3 (July 2002), 582–585.

[JP04] JAMES D. L., PAI D. K.: BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 393–398.

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 399–407.

[JTCW07] JAMES D. L., TWIGG C. D., COVE A., WANG R. Y.: Mesh Ensemble Motion Graphs: Data-driven mesh animation with constraints. *ACM Transactions on Graphics 26*, 4 (Oct. 2007), 17:1—-17:16.

[KCŽO08] KAVAN L., COLLINS S., ŽÁRA J., O'SULLIVAN C.: Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG) 27*, 4 (2008), 105.

[KJ09] KIM T., JAMES D. L.: Skipping steps in deformable simulation with online model reduction. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 123:1–123:9.

[KJP02] KRY P. G., JAMES D. L., PAI D. K.: EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *ACM SIGGRAPH Symposium on Computer Animation* (July 2002), pp. 153–160.

[KLM01] KRYSL P., LALL S., MARSDEN J. E.: Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering 51* (2001), 479–504.

[KMBG08] KAUFMANN P., MARTIN S., BOTSCH M., GROSS M.: Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July 2008), pp. 105–115.

[Kop08] KOPP J.: Efficient numerical diagonalization of Hermitian 3x3 matrices. *Int. J. Mod. Phys.* (2008), 523–548.

[KSJP08] KAUFMAN D. M., SUEDA S., JAMES D. L., PAI D. K.: Staggered projections for frictional contact in multibody systems. *ACM Trans. on Graphics 27*, 5 (Dec. 2008), 164:1–164:11.

[KV08] KIM T., VENDROVSKY E.: DrivenShape: A data-driven approach for shape deformation. In *ACM SIGGRAPH Symposium on Computer Animation* (2008), pp. 49–55.

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 165–172.

[MG03] MOHR A., GLEICHER M.: Building efficient, accurate character skins from examples. *ACM Transactions on Graphics 22*, 3 (July 2003), 562–568.

[MG04] MÜLLER M., GROSS M. H.: Interactive Virtual Materials. In *Graphics Interface 2004* (May 2004), pp. 239–246.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless Deformations Based on Shape Matching. *ACM Trans. on Graphics 24*, 3 (Aug. 2005), 471–478.

[MT92] METAXAS D., TERZOPOULOS D.: Dynamic deformation of solid primitives with constraints. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (July 1992), pp. 309–312.

[RJ07] RIVERS A. R., JAMES D. L.: FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. *ACM Transactions on Graphics 26*, 3 (July 2007), 82:1–82:6.

[Sha90] SHABANA A. A.: *Theory of Vibration, Volume II: Discrete and Continuous Systems*. Springer–Verlag, NY, 1990.

[Sha05] SHABANA A. A.: *Dynamics of Multibody Systems*. Cambridge University Press, NY, 2005.

[SIC01] SLOAN P.-P. J., III C. F. R., COHEN M. F.: Shape by example. In *2001 ACM Symposium on Interactive 3D Graphics* (Mar. 2001), pp. 135–144.

[Smi06] SMITH R.: Open dynamics engine v0.5 user guide, 2006. http://www.ode.org.

[Sta97] STAM J.: Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum 16* (1997).

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically Deformable Models. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (July 1987), pp. 205–214.

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. In *ACM SIGGRAPH Symp. on Computer Animation* (2005), pp. 181–190.

[TW88] TERZOPOULOS D., WITKIN A.: Physically Based Models with Rigid and Deformable Components. *IEEE Computer Graphics & Applications 8*, 6 (Nov. 1988), 41–51.

[WN03] WASFY T., NOOR A.: Computational strategies for flexible multibody systems. *Applied Mechanics Reviews 56*, 6 (2003).

[WPP07] WANG R. Y., PULLI K., POPOVIĆ J.: Real-Time Enveloping with Rotational Regression. *ACM Transactions on Graphics 26*, 3 (July 2007), 73:1–73:9.

[WST09] WICKE M., STANTON M., TREUILLE A.: Modular bases for fluid dynamics. *ACM Trans. on Graphics 28*, 3 (Aug. 2009), 39.