Simulating Articulated Subspace Self-Contact

Yun Teng*
University of California, Santa Barbara

Miguel A. Otaduy†
URJC Madrid

Theodore Kim‡
University of California, Santa Barbara

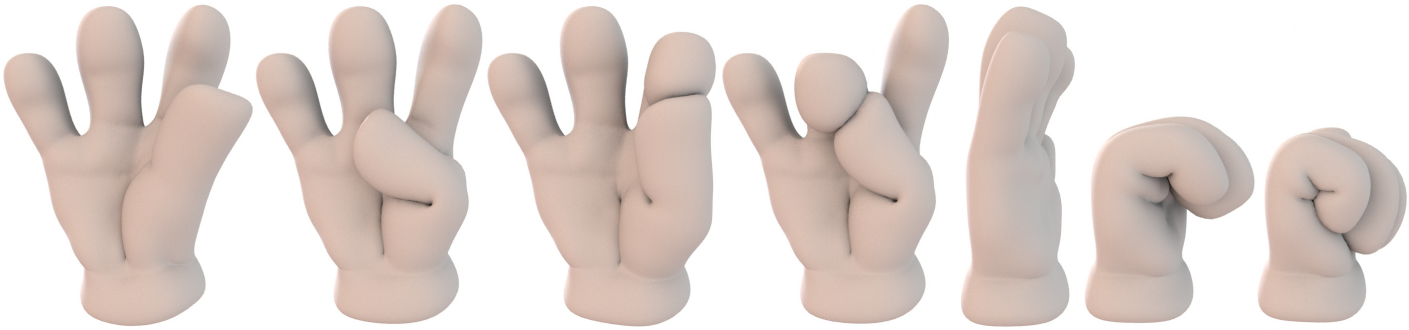


Figure 1: *A hand mesh composed of 458K tetrahedra, running at 5.8 FPS (171 ms), including both self-contact detection and resolution. Our algorithm accelerates the computation of complex self-contacts by a factor of* $5\times$ *to* $52\times$ *over other subspace methods and* $166\times$ *to* $391\times$ *over full-rank simulations. Our self-contact computation never dominates the total time, and takes up at most 46% of a single frame.*

Abstract

We present an efficient new subspace method for simulating the self-contact of articulated deformable bodies, such as characters. Self-contact is highly structured in this setting, as the limited space of possible articulations produces a predictable set of coherent collisions. Subspace methods can leverage this coherence, and have been used in the past to accelerate the collision detection stage of contact simulation. We show that these methods can be used to accelerate the *entire* contact computation, and allow self-contact to be resolved *without looking at all of the contact points*. Our analysis of the problem yields a broader insight into the types of nonlinearities that subspace methods can efficiently approximate, and leads us to design a *pose-space cubature* scheme. Our algorithm accelerates self-contact by up to an order of magnitude over other subspace simulations, and accelerates the overall simulation by two orders of magnitude over full-rank simulations. We demonstrate the simulation of high resolution (100K – 400K elements) meshes in self-contact at interactive rates (5.8 – 50 FPS).

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

Keywords: character simulation, subspace integration, cubature, self-collision detection, self-collision resolution

Links: DL PDF

*yunteng.cs@cs.ucsb.edu
†miguel.otaduy@urjc.es
‡kim@mat.ucsb.edu

1 Introduction

Subspace methods, also known as model reduction, or reduced-order methods, have recently been used to accelerate a variety of simulations in computer graphics, such as deformable bodies [Barbič and James 2005] and fluids [Treuille et al. 2006]. These methods detect temporal redundancies in the simulation during a precomputation stage and use them to construct efficient, low-dimensional subspaces. Simulations are performed in these subspaces at runtime, and routinely yield speedups that are orders of magnitude faster than their "full-rank" counterparts.

This approach is especially appealing for deforming solids, because material properties naturally limit the shapes that a solid model can take on. Thus, it is reasonable to expect that the majority of important deformations can be spanned by a small, "low-rank" subspace. This intuition also extends to articulated bodies, such as character meshes. The global deformation of a character may be complex, but the deformation of an individual limb (e.g. a forearm) is still fairly low-rank. Articulated subspace methods [Kry et al. 2002; Kim and Pollard 2011], which include substructuring and domain decomposition approaches [Barbič and Zhao 2011; Kim and James 2011; Yang et al. 2013] have thus been developed that successfully leverage this coherence.

In the paper, we explore whether this intuition can be applied to self-contact on an articulated mesh. Dense self-contacts involving thousands of collisions routinely arise on these meshes, and can consume more than 90% of the computation time in a single subspace simulation frame. Since these contacts are driven by the articulation and not unpredictable external forces, they display precisely the type of coherent behavior that subspace methods excel at exploiting. However, we are only aware of subspace techniques that accelerate the collision detection stage of contact simulation [James and Pai 2004; Barbič and James 2010]; the contact resolution stage is still computed in a full-rank manner.

We instead present a subspace method that encompasses the entire self-contact simulation. While self-contact is highly coherent, it is also highly non-linear, and necessitates the use of *cubature*, a method that has recently emerged for efficiently handling complex non-linearities in subspace simulations [An et al. 2008]. Our analysis of the self-contact problem improves the understanding of

when cubature succeeds and fails, and provides insights that could be applicable to other problems. Guided by this analysis, we design a *pose-space cubature* scheme that is able to resolve self-contact *without actually detecting and resolving all of the collision points.* Instead, it is only necessary to resolve the contact at a sparse set of "cubature points". Our contributions are as follows:

- We show how to efficiently apply the subspace cubature approach to the problem of self-contact;
- We identify two general conditions that will cause the standard cubature approach to fail: sparse training matrices and excessive discontinuities (e.g. Heaviside functions);
- We propose *pose-space* cubature, which directly addresses these issues and enables efficient subspace self-contact.

2 Related Work

Subspace methods were introduced to graphics by Pentland and Willams [1989] for the simulation of deformable bodies, but were initially applied only to linear materials. They were subsequently extended to support the polynomial systems that arise in deformable body and fluid simulations [Barbič and James 2005; Treuille et al. 2006; Wicke et al. 2009], and most recently to algebraic systems [Stanton et al. 2013]. In all of these cases, the subspace method successfully accelerated the underlying simulation by several orders of magnitude.

The preceding methods work well when the underlying non-linearity can be written as a static tensor and projected as a preprocess. When such a tensor is not available, the *cubature* approach introduced by An et al. [2008] can be applied, as it only requires a method of point-sampling the non-linearity. This approach was successfully applied to fluids [Kim and Delaney 2013] and used to make subspace methods more resilient to external collisions [Harmon and Zorin 2013]. The cubature training stage can be time-consuming, so efficient methods such as importance sampling [Kim and Delaney 2013; Harmon and Zorin 2013] and Hard Thresholding Pursuit (HTP) [von Tycowicz et al. 2013] have been developed. We will also use the cubature approach.

Subspace methods have been used to accelerate various types of contact. James and Pai [2004] showed that a reduced-order triangle inequality could be exploited to construct a very efficient bounding volume hierarchy (a.k.a. a BD-Tree). Self-collision detection has also been accelerated using collision "certificates" [Barbič and James 2010], as well as Bounded-Normal Trees [Schvartzman et al. 2009]. Contact with external objects [Harmon and Zorin 2013] and between two separate deformable objects [Barbič and James 2008] have also been accelerated. All of these works either accelerate only the collision detection stage, or do not address self-contact. In this work, we show that a unified subspace approach can be devised that accelerates the entire self-contact computation pipeline.

More generally, the simulation of deformable bodies is a well-studied problem in computer graphics, and excellent surveys [Gibson and Mirtich 1997; Nealen et al. 2005] and course notes [Sifakis and Barbič 2012] are available. Simulating articulated bodies surrounded by elastic flesh is of particular interest, as they correspond to human and animal characters. In addition to their use in feature animation [Clutterbuck and Jacobs 2010], realistic virtual humans also have numerous medical applications [Hirota et al. 2001]. Thus, many efficient methods have been developed, including co-rotational elasticity and multigrid solvers [Zhu et al. 2010; McAdams et al. 2011; Liu et al. 2013]. These full-rank methods are complementary to our current approach. For example, we use the material and contact model from McAdams et al. [2011] to generate data for our subspace solver. As these solvers become more efficient, our preprocessing stage will proportionally accelerate.

Finally, recent geometric skinning work has yielded impressive results by using implicit surfaces [Vaillant et al. 2013], and also by borrowed ideas from physics-based simulation [Kavan and Sorkine 2012]. We go in the opposite direction and borrow ideas from the geometric skinning [Kavan et al. 2008] and Pose Space Deformation [Lewis et al. 2000] literature to accelerate our physics-based approach. Ours is not the first technique to do this (see e.g. EigenSkin [Kry et al. 2002]), but we specifically design a novel method that uses articulation information to simplify the non-linearity that the cubature scheme must approximate.

3 Simulation Preliminaries

Notation: We will denote vectors in bold lower case, e.g. $\mathbf{x}$, and matrices in bold upper case, e.g. $\mathbf{K}$. Non-linear functions will be denoted with an argument, e.g. $\mathbf{f}(\mathbf{x})$. It is important to distinguish between full-rank and reduced-order quantities, so full-rank quantities will be denoted with an N, and reduced-order with an r. For example, our tetrahedral meshes contain N vertices and $3N$ degrees of freedom (DOFs), but the reduced-order subspace we efficiently simulate within has rank r, where $r \ll N$. We use a tilde to denote a reduced-order quantity, e.g. $\widetilde{\mathbf{x}}$ is a reduced version of $\mathbf{x}$. We will use script to denote sets, e.g. $\mathfrak{C}$ to denote a set of cubature points.

Subspace Projection: We will first review the subspace projection of an elastic deformation problem. We initially describe a quasistatic formulation:

$$\mathbf{f}(\mathbf{x}) = -\mathbf{f}_e - \mathbf{f}_s. \tag{1}$$

Here, $\mathbf{x} \in \mathbb{R}^{3N}$ is a node displacement vector, $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^{3N}$ is a material force response function, and $\mathbf{f}_e \in \mathbb{R}^{3N}$ denotes arbitrary external forces. The $\mathbf{f}_s \in \mathbb{R}^{3N}$ term denotes forces that arise from self-contact, and is the primary quantity that we are concerned with in this paper. We will introduce dynamics later (§5.2), but all of the simulation difficulties are already present in this formulation.

Let us temporarily assume that $\mathbf{f}(\mathbf{x})$ is linear,

$$\mathbf{K}\mathbf{x} = -\mathbf{f}_e - \mathbf{f}_s, \tag{2}$$

where $\mathbf{K} \in \mathbb{R}^{3N \times 3N}$ is a material stiffness matrix. The simulation of this equation can be drastically accelerated by using a subspace basis $\mathbf{U} \in \mathbb{R}^{3N \times r}$. This basis can be obtained using a variety of methods, such as performing PCA on existing full-rank simulation "snapshots" [An et al. 2008], or static analysis of the underlying deformable body mesh [Barbič and James 2005; von Tycowicz et al. 2013]. A much smaller "reduced" version of $\mathbf{K}$ can be computed via projection, $\mathbf{U}^\top \mathbf{K}\mathbf{U} = \widetilde{\mathbf{K}} \in \mathbb{R}^{r \times r}$. Equivalent projections can be performed on the vectors, $\mathbf{U}^\top \mathbf{x} = \widetilde{\mathbf{x}} \in \mathbb{R}^r$, to obtain a final (linear) reduced-order quasistatic equation:

$$\widetilde{\mathbf{K}}\widetilde{\mathbf{x}} = -\mathbf{U}^\top \mathbf{f}_e - \mathbf{U}^\top \mathbf{f}_s. \tag{3}$$

Once all the projections have been performed, $\widetilde{\mathbf{x}}$ can be solved for very efficiently, as the main computational challenge in the full-rank case is a linear solve with respect to $\mathbf{K}$. Now $\widetilde{\mathbf{K}}$ can be solved very quickly, or even explicitly inverted once as a pre-process. This is the projection-based version of subspace simulation [Pentland and Williams 1989].

Subspace Cubature: Most realistic material models are not linear, so we now drop the assumption that $\mathbf{f}(\mathbf{x}) = \mathbf{K}\mathbf{x}$. Important cases such as the St. Venant-Kirchhoff material model can still be written

as polynomials, which permit the projection approach to be applied, albeit using higher order tensors [Barbič and James 2005].

However, a variety of other materials cannot be written in this form, particularly the co-rotational material model we use in this paper [McAdams et al. 2011]. The main issue is that per-element polar decompositions must be performed, which are non-trivial to formulate as a tensor. The extensions introduced by Stanton et al. [2013] do not appear to provide any assistance in this case.

Fortunately, the *cubature* approach to subspace simulation can be used to efficiently compute arbitrary non-linear $\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}})$ terms [An et al. 2008]. Instead of projecting a tensor, the cubature approach approximates the underlying function using a carefully weighted set $\mathfrak{C}$ of C "cubature points":

$$\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}) \approx \sum_{i=1}^{C} w_i \cdot \mathbf{U}_i^\top \mathbf{f}_i(\mathbf{U}_i \widetilde{\mathbf{x}}). \tag{4}$$

Here, w_i is a scalar cubature weight applied to sample i, $\mathbf{U}_i \in \mathbb{R}^{12\times r}$ is the rows of $\mathbf{U}$ that correspond to the vertices of tet i, and $\mathbf{f}_i$ is a point-sampled version of the material force function that computes the material forces on the vertices of the ith tet.

Connections between Eqn. 4 and the compressive sensing literature have been observed [von Tycowicz et al. 2013], and this perspective provides some intuition. Similar to how the Fourier transform of a Dirac delta yields a function with global support, transforming a point sample $\mathbf{f}_i(\mathbf{U}_i\widetilde{\mathbf{x}})$ into the subspace $\mathbf{U}$ also yields a global function. The question is whether a global function $\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}})$ can be approximated by projecting a small number of samples. The subspace modes of $\mathbf{U}$ were constructed by performing PCA, which succinctly parameterizes all previously seen examples of $\mathbf{f}(\mathbf{x})$, so it is reasonable to expect that within this subspace, some succinct version $\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}})$ can be discovered.

The cubature approach has been successfully applied to a wide variety of non-linearities, including several non-linear material models [An et al. 2008], and the quadratic term in the inviscid Euler equations [Kim and Delaney 2013]. In our current work, we have found that it can be applied to the material model of interest [McAdams et al. 2011], and more importantly, to self-collisions.

The Subspace Self-Collision Problem: We now have the subspace simulation equation:

$$\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}) = -\mathbf{U}^\top \mathbf{f}_e - \mathbf{U}^\top \mathbf{f}_s. \tag{5}$$

The $\widetilde{\mathbf{f}}(\widetilde{\mathbf{x}})$ term can be computed efficiently using Eqn. 4. Efficient computation of external forces is the focus of previously discussed works [Barbič and James 2008; Harmon and Zorin 2013], so the question we are interested in is: Can we compute $\mathbf{U}^\top \mathbf{f}_s$ efficiently?

The methods used for external forces (i.e. $\mathbf{U}^\top \mathbf{f}_e$) could be applied, but this would discard much of the knowledge we have of $\mathbf{f}_s$. The variety of possible self-contact forces is largely constrained by the articulation limits on the skeleton, and while extreme self-contact configurations are possible (e.g. the character being crushed in a trash compactor) the vast majority of forces will repetitively occur at a predictable set of surface vertices. We will use this knowledge to design an efficient cubature scheme for $\mathbf{U}^\top \mathbf{f}_s$ in Section 4.

Penalty Force Formulation: Like many recent works [Tang et al. 2012; McAdams et al. 2011; Harmon et al. 2009], we use a penalty force formulation to resolve contacts. For each penetrating surface triangle and vertex, $(\mathbf{t}_p, \mathbf{v}_p)$, we find the surface point $\mathbf{v}_s$ on the triangle $\mathbf{t}_p$ that is closest to $\mathbf{v}_p$ and apply a force to $\mathbf{v}_p$,

$$\mathbf{f}_p = k_c \cdot k_a \cdot \mathbf{N}(\mathbf{v}_s - \mathbf{v}_p), \tag{6}$$

where k_c is a constant spring stiffness and k_a is the average of the area of $\mathbf{t}_p$ and the area of the surface triangles in the one-ring surrounding $\mathbf{v}_p$. The $\mathbf{N} \in \mathbb{R}^{3\times3}$ anisotropy term is from McAdams et al. [2011], and defined as

$$\mathbf{N} = (1-\alpha)\mathbf{n}\cdot\mathbf{n}^\top + \alpha\mathbf{I}, \tag{7}$$

where $\alpha \in [0,1]$, $\mathbf{I} \in \mathbb{R}^{3\times3}$ is an identity matrix, and $\mathbf{n}$ is the outward unit contact normal, $\mathbf{n} = \frac{(\mathbf{v}_s - \mathbf{v}_p)}{|\mathbf{v}_s - \mathbf{v}_p|}$. The force at each ith vertex of the triangle $\mathbf{t}_p$ is:

$$\mathbf{f}_i = -\beta_i \cdot k_c \cdot k_a \cdot \mathbf{N}(\mathbf{v}_s - \mathbf{v}_p), \tag{8}$$

where β_i is the barycentric coordinate associated with triangle vertex i and $\mathbf{v}_s$. Section 5.1 contains further details on collisions.

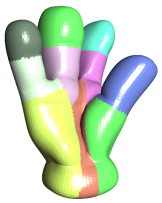
Domain Decomposition: As is common in articulated subspace simulations [Yang et al. 2013; Barbič and Zhao 2011; Kim and James 2011], we will assume that the mesh has been partitioned into subdomains. We used a simple nearest-neighbor strategy to associate each tet with the nearest bone, but our algorithm is agnostic to the partitioning method, so other methods could be used. Each subdomain has its own subspace basis and the interfaces are coupled using penalty-based forces.

4 A Self-Collision Cubature Scheme

Self-contacts on articulated meshes occur mainly due to the motion of the underlying skeleton. We still start by directly applying the cubature approach to these self-collisions. The algorithm will produce unacceptable results, but motivate a more sophisticated scheme inspired by Pose Space Deformation [Lewis et al. 2000].

4.1 A Direct Cubature Scheme

For expository purposes, we will focus on a single domain, x, and examine the collisions that arise between itself and another domain, y. We emphasize that these include *intra*-domain collisions between the surface of x and itself that arise due to the motion of y, not just the collisions between x and y. An example of our final algorithm handling such collisions is inset on the left. The hand contains 10 domains, visualized by different colors. Note that the red domain in the middle of the palm is colliding with itself. We denote the full-coordinate ranks of x and y as N_x and N_y, and their reduced ranks as r_x and r_y. Each domain will have its own basis, $\mathbf{U}_x$ and $\mathbf{U}_y$.



The $\mathbf{f}_s$ self-collision term for domain x is intrinsically a force. Since the cubature approach was successfully applied to the other force terms such as the $\mathbf{f}(\mathbf{x})$ material force term, it is reasonable to expect that it will be able to generalize to this term as well. Within this context, a single cubature point p is represented by the self-collision force, $\mathbf{f}_s(\mathbf{t}_p, \mathbf{v}_p) \in \mathbb{R}^{N_x+N_y}$, with only 12 non-zero entries arising from the collision of a single vertex, $\mathbf{v}_p \in \mathbb{R}^3$, with a triangle, $\mathbf{t}_p \in \mathbb{R}^9$. Note that in this context, a "cubature point" no longer corresponds to a literal spatial point or tetrahedron, but rather a geometric pair. With this definition established, we can compute a self-collision cubature scheme using the same training algorithm from An et al. [2008].

At a high level, the training problem we are trying to solve is as follows. We have a set of self-collision forces that were generated from a large number of contact pairs, and would like to approximate these forces by instead using a weighted subset of these pairs. The primary question of interest is: how many pairs are needed to achieve a desired accuracy?

We will limit ourselves to training details that will be relevant when later examining why this direct scheme fails. We have a set of T training snapshots (i.e. example deformations), which we obtain by running a series of full-rank simulations. Each t-th snapshot possesses a self-contact force vector, $\mathbf{f}_s^t \in \mathbb{R}^{N_x+N_y}$. A reduced version of this vector can be obtained via projection, $\widetilde{\mathbf{f}}_s^t = \mathbf{S}^\top \mathbf{f}_s^t$, where $\mathbf{S} \in \mathbb{R}^{(N_x+N_y)\times(r_x+r_y)}$ and is constructed as the block matrix:

$$\mathbf{S} = \begin{bmatrix} \mathbf{U}_x & 0 \\ 0 & \mathbf{U}_y \end{bmatrix}. \tag{9}$$

The goal of the training stage is to locate a small cubature set $\mathfrak{C}$ consisting of C collision pairs that, when evaluated,

$$\widetilde{\mathbf{f}}_s^t \approx \sum_{p=1}^{C} w_p \cdot \mathbf{S}_p^\top \mathbf{f}_s(\mathbf{t}_p^t, \mathbf{v}_p^t), \tag{10}$$

approximate all T self-collision forces to some desired degree of accuracy. Here, $(\mathbf{t}_p^t, \mathbf{v}_p^t)$ denotes a collision pair deformed according to the displacement vector from the corresponding snapshot t, and $\mathbf{S}_p$ denotes a concatenation of the rows in $\mathbf{S}$ that correspond to $\mathbf{v}_p$ and the vertices in $\mathbf{t}_p$.

As with material forces (Eqn. 4), there is reason to believe that a sparse approximation exists. For example, imagine that only a single self-collision example had been observed, and its $\mathbf{f}_s$ was added as the final column in $\mathbf{U}_x$. A single cubature point would suffice, provided its projection produced a vector composed entirely of zeros, except for a final non-zero component. The w_p would account for any scalar discrepancy, and the global force would be effectively represented by a single point.

Promising candidates for $\mathfrak{C}$ can be located in a variety of ways, including a greedy search [An et al. 2008], importance sampling [Kim and Delaney 2013; Harmon and Zorin 2013], or HTP [von Tycowicz et al. 2013]. We used importance sampling in this work. Once candidates are located, a Non-Negative Least Squares (NNLS) solve assigns each candidate a weight (w_p in Eqn. 10). First, the projected self-collision force produced by each p-th candidate at each t-th snapshot is computed,

$$\begin{bmatrix} \mathbf{S}_p^\top \mathbf{f}_s(\mathbf{t}_p^1, \mathbf{v}_p^1) \\ \mathbf{S}_p^\top \mathbf{f}_s(\mathbf{t}_p^2, \mathbf{v}_p^2) \\ \cdots \\ \mathbf{S}_p^\top \mathbf{f}_s(\mathbf{t}_p^T, \mathbf{v}_p^T) \end{bmatrix} = \mathbf{g}_p. \tag{11}$$

These columns are concatenated to form the $\mathbf{A}\mathbf{w} = \mathbf{b}$ system

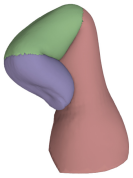
$$\begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \dots & \mathbf{g}_C \end{bmatrix} \mathbf{w} = \begin{bmatrix} \left(\widetilde{\mathbf{f}}_s^1\right)^\top & \left(\widetilde{\mathbf{f}}_s^2\right)^\top & \dots & \left(\widetilde{\mathbf{f}}_s^T\right)^\top \end{bmatrix}^\top, \tag{12}$$

where $\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_C \end{bmatrix}^\top$ is the vector of weights that is solved for. This system can then be sent to any standard NNLS solver [Chen and Plemmons 2007].

4.2 Analysis of Negative Results

We found that the cubature sets discovered by this direct cubature method are not immediately useful. In order to converge to an acceptable level of relative error of 1% to 5%, between 30% and 40% of all of the collision pairs in the training snapshots had to be added to the cubature set. These sets clearly did not have $O(r)$ cardinality, and did not significantly accelerate self-contact computation. Alternate candidate selection approaches such as HTP [von Tycowicz et al. 2013] may be able to decrease the cardinality by a small constant, but the asymptotic performance is unlikely to change.

Infrequent events in the training set were captured poorly.



In Figure 4 for example, the top pad of the finger (left image, in blue) comes into contact with the bottom third of the finger (left image, in red) only in the final frames of our training set. Since these forces do not appear in many snapshots, the NNLS solver treats them as unimportant, and the final cubature set approximates them poorly. While it might be possible to re-weight these examples, such an approach would essentially second-guess the PCA and NNLS solvers, which seems unappealing. Clearly, this naïve application of the cubature approach is not sufficient.

The intuition from §4.1 breaks down because it applied to a single self-collision, $\widetilde{\mathbf{f}}_s^t$, not the set of all self-collisions. While a single approximation may be sparse, the set of all approximations may be spatially disjoint, since the contact regions can be disjoint for two radically different poses. The union of the sparse approximations can then be dense if sparsity structures do not share any commonality. We can quantify this intuition as follows.

Sparsity of the training matrix A: The training column $\mathbf{g}_p$ for any collision pair is likely to contain mostly zeros, since the pair is likely in collision for only a small number of snapshots. Therefore, many of the $O(T)$ rows in the $\mathbf{A}$ matrix (Eqn. 12) will also consist solely of zeros. Unless a collision pair is added to the cubature set that specifically introduces non-zero entries into these rows, their relative errors will be 100%. This explains why the error only became acceptable for cubature sets of size $O(N)$, or more precisely $O(P)$, where P is the total number of collision pairs in all of the training snapshots; it is threshold where the number of non-zero rows in $\mathbf{A}$ had been sufficiently increased. We can conclude from this that *a sparse training matrix will produce a dense cubature set.*

Difficulty of learning the non-linear function: Eqn. 6, the non-linear force being learned, appears to be essentially a cubic function. However, the function actually contains an additional non-linearity. We can write a more general force term,

$$\mathbf{f}_p(\mathbf{t}_p, \mathbf{v}_p) = H(\mathbf{t}_p, \mathbf{v}_p) \cdot k_c \cdot k_a \cdot \mathbf{N}(\mathbf{v}_s - \mathbf{v}_p), \tag{13}$$

where $H(\mathbf{t}_p, \mathbf{v}_p)$ is a Heaviside function defined as:

$$H(\mathbf{t}_p, \mathbf{v}_p) = \begin{cases} 1 & \text{if } \mathbf{t}_p \text{ and } \mathbf{v}_p \text{ collide} \\ 0 & \text{otherwise} \end{cases}. \tag{14}$$

This new term significantly increases the non-linearity, as its dependence on $\mathbf{x}$ indirectly introduces the complexity of the material model from Eqn. 1. The binary nature of the term also shows that while the force is C^0 continuous, it contains a C^1 discontinuity. We will need to address the complexity introduced by this term.

4.3 A Pose-Space Cubature Scheme

The preceding analysis led us to design a method that leverages the same intuition that underlies Pose Space Deformation (PSD) [Lewis et al. 2000]. Instead of trying to capture a difficult non-linearity in its entirety, we interpolate over a sparse set of solutions where the non-linearity is known to be efficiently resolvable. In the case of PSD, a set of artist-sculpted examples are interpolated. In our case, we interpolate over a set of *per-pose* cubature schemes.

Per-Pose Cubature: Instead of computing a *single* cubature scheme that fits a large, sparse training matrix, we compute *multiple* cubature schemes that each fit small, dense matrices. At a high level, the primary difference between this training regimen and the previous one is that we are no longer trying to find a single cubature scheme that can approximate every self-collision force in the training set. Instead, we compute one cubature scheme for every

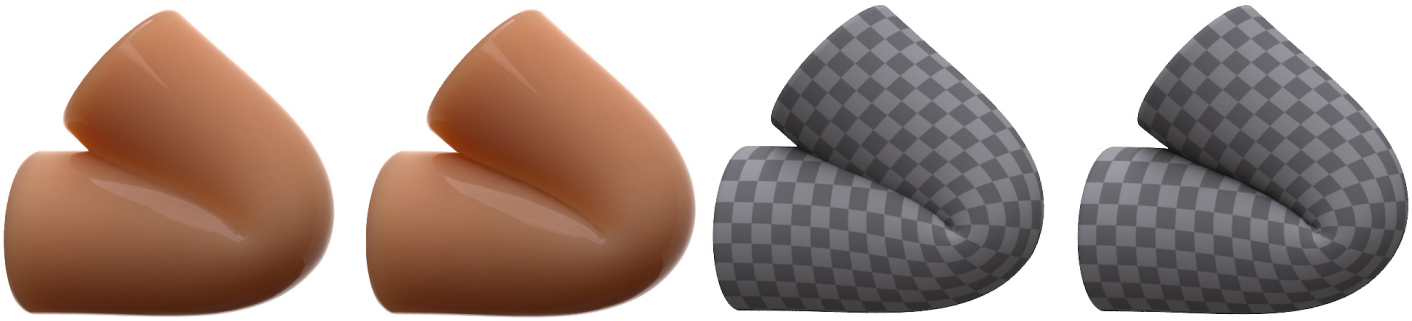


Figure 2: *A cylinder articulated with a single joint.* **Left in each pair:** *Without self-contact.* **Right in each pair:** *With self-contact. Penetrations obscure the characteristic crease along the contact, especially in the subsurface scattering rendering. We accelerate self-contact by* $52\times$ *and only take up 46% of the total computation time instead of 96%.*

example pose in the training set. If a pose is encountered at runtime that was not in the original training set, we can use a PSD-like process to interpolate between the nearest schemes.

The simplest training matrix that we can construct that is guaranteed to be dense is the one for a single training snapshot. For example, if we define the projected force for the 1^{st} snapshot for the collision pair p as $\mathbf{S}_p^\top \mathbf{f}_s(\mathbf{t}_p^1, \mathbf{v}_p^1) = \mathbf{g}_p^1$, we can write the training matrix:

$$\begin{bmatrix}\mathbf{g}_1^1 & \mathbf{g}_2^1 & \dots & \mathbf{g}_C^1\end{bmatrix} \mathbf{w}^1 = \widetilde{\mathbf{f}}_s^1. \tag{15}$$

We abbreviate this system to $\mathbf{A}^1\mathbf{w}^1 = \widetilde{\mathbf{f}}_s^1$. Note that the right hand side now contains a single self-contact force sample, and that the training matrix dimensions are $\mathbf{A}^1 \in \mathbb{R}^{r\times C}$ instead of $\mathbf{A} \in \mathbb{R}^{rT\times C}$, essentially making $\mathbf{A}^1$ a subset of rows from $\mathbf{A}$. This training matrix will automatically be dense. All of the cubature selection strategies (greedy, importance sampled, or HTP) project potential candidates onto the residual vector $\mathbf{r} = \mathbf{A}^1\mathbf{w}^1 - \widetilde{\mathbf{f}}_s^1$. A collision pair that produced a zero-valued training column would also produce a zero-valued projection, and would therefore never be considered promising enough to be added to the cubature set. As predicted, we found that the training stage now produced cubature sets with $O(r)$ cardinality for each input snapshot (Figure 3).

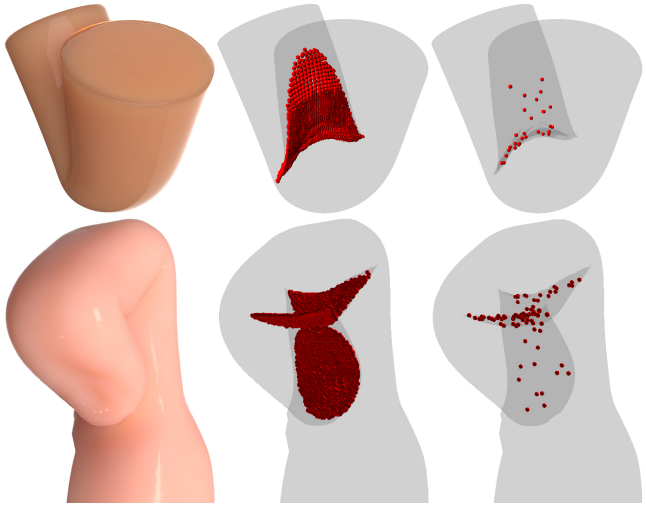


Figure 3: Left: *Final frames from Figs. 2 and 4.* **Middle:** *Dense, full-rank contact points.* **Right:** *Our cubature scheme only needs to detect and resolve a sparse subset of these points.*

Cubature Interpolation: We now have T cubature sets to interpolate at runtime. While the direct cubature scheme contained a difficult-to-learn Heaviside function, we now have the opposite problem that this non-linearity has been aggressively pre-processed away. At runtime, each cubature set assumes that the Heaviside functions have already been resolved, and all that remains is for the contact forces (Eqns. 6 and 8) to be computed. This is not an issue if the character only takes on poses that are exactly the same as the training snapshots, but much of the appeal of subspace methods is that they can generalize to motions that are similar to, but distinct from, those in the training set. A method is needed for interpolating between cubature sets when a pose is encountered that was not seen during training. During this interpolation, $H(\mathbf{t}_p, \mathbf{v}_p)$ is evaluated at each cubature point, as it is unknown whether the collision occurs at this intermediate pose.

We perform the interpolation as follows. At runtime, we locate the closest cubature set $\mathfrak{C}_t$ of all T cubature sets, by computing the distance between some feature vector of the current pose, $\mathbf{q}_c \in \mathbb{R}^6$, and the corresponding $\mathbf{q}_i \in \mathbb{R}^6$ for each of the cubature sets:

$$t = \arg\min_{i\in T} \left(\text{dist}(\mathbf{q}_c, \mathbf{q}_i)\right). \tag{16}$$

where "dist" denotes a distance function. We have some freedom in designing both "dist" and the pose vectors, $\mathbf{q}$. If domains x and y are connected by a joint, quaternions could be used ($\mathbf{q} \in \mathbb{R}^4$), and a natural distance measure would be $1 - \|\mathbf{q}_c \cdot \mathbf{q}_i\|$. However, if x and y are not adjacent, a translation needs to be added to $\mathbf{q}$. This complicates the distance measure, because while quaternions are normalized, translations are not, and a large value can spuriously dominate the distance measurement. Separate weightings could be added to rotation and translation, but such a scheme would be ad hoc.

We use a PCA method [Shabana 1990] to address this issue. In an offline stage, for each training snapshot, we transform the surface positions of domain y into x's local coordinates system. We perform PCA on these positions and extract the six most significant components, $\mathbf{U}_r$. For an unconstrained mesh, these components will have principal values of zero, as they correspond to the zero-energy, infinitesimal rigid transformation modes of the mesh (see e.g. [Idelsohn and Cardona 1985], Appendix D in [Barbič and James 2005], or §3.1 in [O'Brien et al. 2002]). The subspace coordinates $\mathbf{q}_i$ for each training snapshot i are computed and stored along with $\mathbf{U}_r$. The distance between two poses c and i can now be defined as $\text{dist}(\mathbf{q}_c, \mathbf{q}_i) = ||\mathbf{q}_c - \mathbf{q}_i||_2$. At run time, we again transform the surface positions of y into the local coordinates of x, and project it using $\mathbf{U}_r$ to obtain $\mathbf{q}_c$.

We find the two nearest neighbors from Eqn. 16 by using a brute force search. The subspace forces from the two schemes are then interpolated based on their relative distances. Because the number of snapshots T is small in all of our examples (see Table 1), the brute force search takes a negligible amount of time. If T were larger and created a bottleneck, more efficient methods [de Berg et al. 2008] could be used.

Discussion: This approach was inspired by Pose Space Deformation, but the resulting algorithm is fairly different. We use a linear blend instead of radial basis functions, as we found they were more appropriate for our non-linearities. Each cubature scheme is localized to a pair of domains, which is reminiscent of Weighted PSD [Kurihara and Miyata 2004], and suggests that a non-partitioned algorithm that can be applied to monolithic subspace methods [Kry et al. 2002; Kim and Pollard 2011] should also possible. The final outputs of our algorithm are generic subspace forces and stiffnesses, which do not fundamentally depend on the existence of a spatial decomposition. We leave a more in-depth investigation into these issues and their relationship with PSD as future work.

5 Implementation and Results

5.1 Implementation

Solvers: All of our simulations used implicit Newmark as the time discretization. We used the interior point optimizer IPOPT [Wächter and Vigerske 2005; Wächter 2002] as our non-linear solver, as we found its wall-clock time to be faster than Newton-Raphson. In the inner loops of the full-rank and reduced-rank solves, we respectively used the highly tuned HSL solvers `MA86` [Hogg and Scott 2010] and `MA57` [Duff 2004]. We found that these were faster than conjugate gradient, and outperformed even an improved Incomplete Cholesky preconditioner [Jones et al. 1995].

Hardware: All simulations were run on a 12-core, 2.66 Ghz Mac Pro with 96 GB of RAM. Our own code and the HSL solvers both leveraged OpenMP for parallelization. All of the full-rank simulations ran on 12 threads. For each subspace example, we manually optimized the number of threads to fit the number of domains.

Basis Construction: To obtain meaningful forces, we explicitly enriched the subspace basis $\mathbf{U}$ with the self-contact forces. We computed the self contact forces for a variety of training poses, and perform a PCA on the resulting $\mathbf{f}_s$ vectors. We then concatenate the results to $\mathbf{U}$ and re-orthogonalize using modified Gram-Schmidt. We found it necessary to add the self-contact force vectors to the *front* of $\mathbf{U}$. Otherwise, the locality of the forces would become entangled with the global support from other columns in $\mathbf{U}$.

Collision Detection: During full-rank simulation, we used DeformCD [Curtis et al. 2008] for collision detection. This algorithm uses a bounding volume hierarchy, specifically axis-aligned bounding boxes (AABBs), for the broad phase, and the Representative Triangles during the narrow phase.

For subspace collision detection, we also used AABBs for the broad phase, and then brute-force point-tet intersection tests for the narrow phase. We built two bounding volume hierarchies (BVH) for each domain, one for the cubature vertices and one for the tetrahedra. At each timestep, the cubature vertices from one domain x was tested against *all* of the tetrahedra from another domain y (for intra-domain collisions, $x = y$). We could have only performed collisions against the triangles $\mathbf{t}_p$ from the training set, but we found this to be too conservative for high-resolution meshes, as the cubature vertex $\mathbf{v}_p$ could easily penetrate a nearby triangle instead. The cubature weights were still effective, since nearby triangles generate similar penalty forces. We also attempted to use BD-Trees [James and Pai 2004], but found our BVH was faster. The deformations required many leaf-level updates, and the $\mathbf{U}\tilde{\mathbf{x}}$ multiply at each BD-Tree node update quickly became a bottleneck. As we were simulating high resolution meshes, we assumed that edge-edge and vertex-edge contacts are negligible, but our algorithm could be extended to handle these case as well.

As anticipated, the cardinality of the cubature sets was much smaller than the complete set of contact points, and reduced the number of necessary collision tests considerably (Table 2). The cubature schemes sufficiently accelerated the simulation that collision detection dominated the running time ($\sim$90%). We experimented with diminishing this bottleneck by using a low-resolution proxy of the tetrahedral mesh for the broad phase of subspace collision detection. This accelerated the phase significantly, and as the high-resolution mesh is queried during the narrow phase, the impact on the final results was minimal. To facilitate comparisons, all of the timings given in Table 2 and Figs. 6 and 7 use this proxy geometry. We did not use the low-resolution proxy during the full-rank precomputation, as collision detection was not the main bottleneck, and instead took between 16-27% of the running time. The increase in computational cost instead appeared in the form of more complex non-linear solves.

Collision Resolution: We called the same collision resolution code for both the reduced- and full-rank simulations. For each penetrating vertex $\mathbf{v}_p$ and penetrated tet $\mathbf{h}$, we used barycentric interpolation to transform $\mathbf{v}_p$ to its position inside of the rest-pose version of $\mathbf{h}$, $\overline{\mathbf{v}}_\mathbf{h}$. We obtained the surface point $\overline{\mathbf{v}}_s$ and surface triangle $\mathbf{t}_p$ closest to $\overline{\mathbf{v}}_\mathbf{h}$ using the signed distance field of the rest pose [Hirota et al. 2001; Teran et al. 2005]. Barycentric interpolation was used to obtain the deformed surface position, $\mathbf{v}_s$. The resulting $\mathbf{v}_s$ and $\mathbf{v}_p$ were then sent to Eqns. 6 and 8.

Material Model: In all of our examples, we used the co-rotational material, along with indefiniteness correction, from McAdams et al. [2011]. Our simulations were performed over tetrahedral meshes instead of hexahedral meshes, so we did not find it necessary to perform their stabilization step, as the deformation nullspace of a tetrahedron is much smaller than that of hexahedron (3 instead of 15). The cubature approach successfully approximated the co-rotational material model with minimal intervention.

Example	Full-Rank Time Per Frame	Reduced-Rank Time Per Frame	Speedup
Cylinder	7.82 s	20 ms	391×
Finger	15.90 s	53 ms	300×
Arm	20.86 s	77 ms	271×
Hand	28.45 s	171 ms	166×

Table 1: *Algorithm performance compared to full-rank solves.*

5.2 Results

In our examples, we sparsely sampled the articulation by computing T samples in joint space (Table 1). For each pose, we ran a full-rank quasistatic simulation with self-contact detection and resolution, and a pose-space cubature scheme was computed to within 5% training error. In all cases, the subspace simulations ran *two orders of magnitude* faster than full-rank solves (Table 1).

Cylinder: We deformed an articulated cylinder containing 118,389 tetrahedra into a highly colliding state in Fig. 2. The mesh contained 2 domains, and was trained on $T = 12$ examples for a final rank of $r = 20$. Without cubature, the subspace simulation spent *up to 96%* of its frame computation time detecting and resolving collisions. With it, self-contact computation *never exceeded 46%*

Example	Using Reduced Self-Contact?	Mean SC Time Per Frame	Max SC Time Per Frame	Max # of Collisions	Collision Complexity Reduction	Mean Speedup	Max Speedup
Cylinder	No	114 ms	415 ms	4127			
Cylinder	**Yes**	7 ms	8 ms	15	275×	16×	52×
Finger	No	168 ms	517 ms	4930			
Finger	**Yes**	17 ms	26 ms	81	61×	10×	20×
Arm	No	182 ms	407 ms	1264			
Arm	**Yes**	26 ms	37 ms	74	17×	7×	11×
Hand	No	59 ms	782 ms	3025			
Hand	**Yes**	32 ms	155 ms	445	7×	2×	5×

Table 2: *Performance of our self-contact (SC) algorithm compared to a subspace simulation that does not use self-contact cubature.*

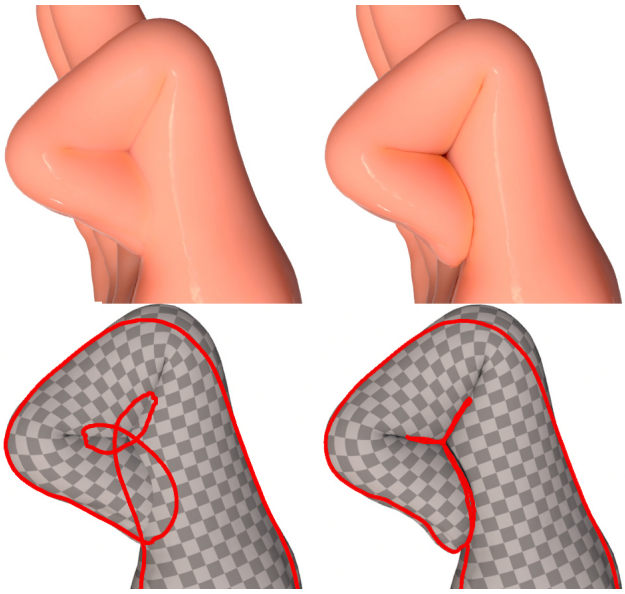


Figure 4: Left column: *Without self-contact.* **Right column:** *With self-contact. A finger is shown with subsurface scattering (top), and with checkerboard texturing and a highlighted cross-section (bottom). Extreme penetrations occur without our self-collision cubature; the full extent can be seen in the cross-section. As with Figure 2, the characteristic crease from the contact is far less visible.*

(Figure 6). Self-contact in the most highly colliding frame was *accelerated by a factor of 52×*.

Finger: We simulated the complex self-collision of a finger containing 248,869 tetrahedra in Fig. 4. The mesh contained 3 domains, and trained on $T = 17$ examples for a final rank of $r = 60$. This example shows our pose-space cubature scheme capturing infrequent contacts between initially non-adjacent domains; the exact configuration that defeated the direct cubature scheme (§4.2). Without cubature, self-contact consumed *up to 96%* of the frame time, but the cubature version *never exceeded 44%*. The most highly colliding frame was *accelerated by a factor of 20×*.

Arm: We simulated the collision of an arm with a body on a mesh composed of 171,492 tetrahedra (Fig. 5). The mesh contained 5 domains, and trained on $T = 27$ examples for a final rank of $r = 161$. All of the initial training data was quasistatic. Without cubature, self-contact took *up to 89%* of the time, while our version *never exceeded 43%*. The most highly colliding frame was *accelerated by 11×*.

This example shows self-contacts resolved over a wide spatial extent, and subspace dynamics added as a post-process. Dynamics were activated using standard methods, i.e. by adding reduced-order mass and damping terms to the left hand side of Eqn. 5: $\widetilde{\mathbf{M}}\ddot{\widetilde{\mathbf{x}}} + \widetilde{\mathbf{C}}\dot{\widetilde{\mathbf{x}}} + \widetilde{\mathbf{f}}(\widetilde{\mathbf{x}}) = -\mathbf{U}^\top \mathbf{f}_e - \mathbf{U}^\top \mathbf{f}_s$. Newmark integration (see Appendix C of Barbič and James [2005]) was then applied. Other methods could also have been used, as our method is agnostic to the underlying time integration and domain decomposition methods.

Hand: We simulated a hand containing 458,071 tetrahedra in a variety of poses. The mesh contained 10 domains, and was trained on $T = 120$ examples for a final rank of $r = 300$. The self-contacts in this example were sparser and less persistent than those in the other examples, so the performance is noisier (Fig. 6). Without cubature, self-contact took *up to 72%* of computation, while our computation *never exceeded 42%*. The most highly colliding frame was *accelerated by 5×*. We also show an example of a collision very different from those seen during cubature training in the video. As expected, the collision is missed entirely, and the deformations begin to exhibit subspace artifacts. However, the simulation remains stable.

Discussion: Predictably, the largest speedups occurred when the largest number of primitives were in self-collision (Table 2). Faster collision detection could improve the performance of the simulations without cubature, but these methods would apply equally well to the cubature versions, so we expect that the relative performances would remain similar.

Direct comparisons to other methods are difficult due to implementation and hardware specifics, but rough, order-of-magnitude comparisons are possible. McAdams et al. [2011] perform full-rank simulations of equivalent geometric complexity as ours (∼100K elements), but our performance is a clear improvement (5 s vs. 53 ms). Their algorithm is complementary and could be used to accelerate our precomputation. Other, faster methods use geometries that are orders of magnitude simpler, e.g. 4160 [Liu et al. 2013] and 8619 [Kim and Pollard 2011] elements. Our method offers clear advantages for both computational and geometric complexity.

6 Conclusion

We have shown that the cubature approach can be used to accelerate self-contact computation in articulated simulations. The standard approach can fail if the non-linearity of the underlying function is too severe, so we formulated a pose space variant to address this limitation. Interestingly, we found that the sparsity of the cubature training matrix serves as a useful heuristic for determining if the underlying function is "too non-linear."

As observed by others [Harmon and Zorin 2013], a comprehensive theory of cubature sampling is still an open problem. We did not present such a theory here, but we hope our results and observations can assist in the formation of such a theory in the future. Finally, the high resolutions supported by our algorithm enable the efficient

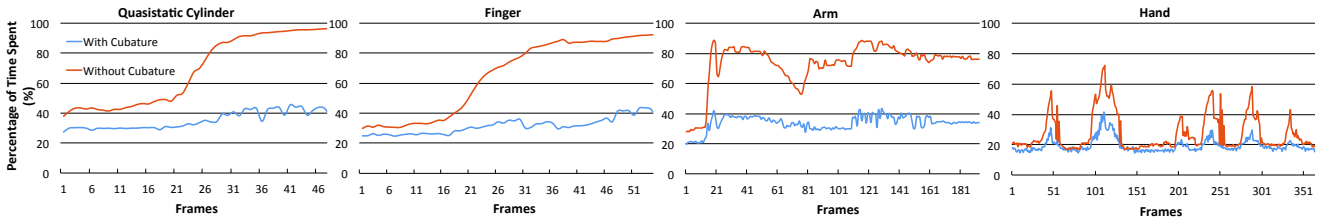


Figure 6: *Percentage of time the subspace simulation spends in self-contact detection and resolution, with and without cubature. Without cubature, the time spent routinely exceeds 90%. Our method consumes a maximum of 46%, bringing it to parity with the rest of the simulation.*

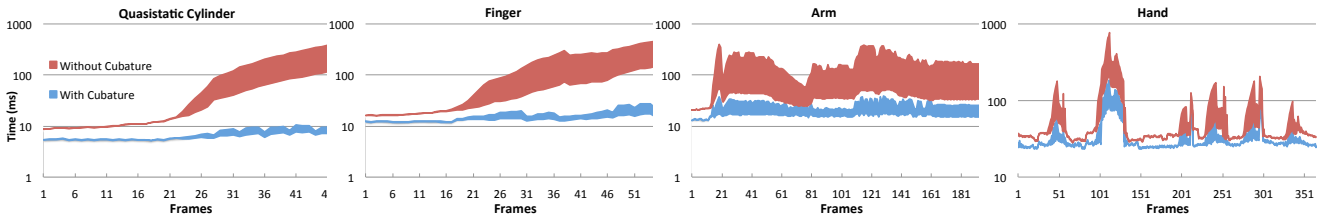


Figure 7: *Time spent in self-collision detection and resolution, with and without cubature. The lower boundary of each shaded area is the time spent in self-collision detection, and the upper boundary shows the total time when resolution is added. Note that we have used a log scale, because otherwise the additional time added by collision resolution when using cubature would be difficult to see.*

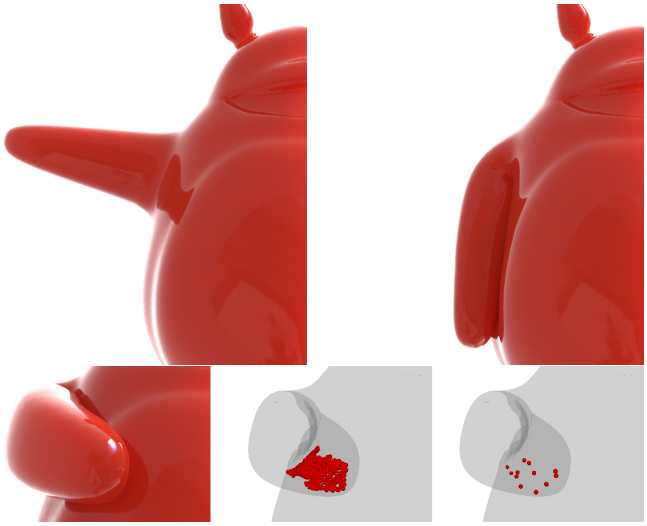


Figure 5: *We can quickly simulate subspace dynamics, even if the input data was quasistatic. The character's arm begins outstretched (top left), but later hits the body (top right), causing it to shake. Please see the video to view the overall dynamics. As in Fig. 3, the cubature scheme effectively sparsifies the contact (bottom row).*

simulation of high-frequency deformations in inhomogeneous materials. However, spatially tuning material parameters to achieve specific effects remains a challenge.

Acknowledgements: The authors would like to thank Eftychios Sifakis for advice on material parameters and Ai Takahashi for building the model in Fig. 5. YT and TK are supported by a National Science Foundation CAREER award (IIS-1253948). MO was partially funded by the European Research Council (ERC-2011-StG-280135 Animetrics) We acknowledge rendering support from the Center for Scientific Computing from the CNSI, MRL: an NSF MRSEC (DMR-1121053), Hewlett-Packard, and NSF CNS-0960316. 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.

References

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

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

BARBIČ, J., AND JAMES, D. L. 2008. Six-DoF haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Transactions on Haptics 1*, 1 (jan.-june), 39–52.

BARBIČ, J., AND JAMES, D. L. 2010. Subspace self-collision culling. *ACM Trans. Graph. 29*, 4 (July), 81:1–81:9.

BARBIČ, J., AND ZHAO, Y. 2011. Real-time large-deformation substructuring. *ACM Trans. on Graphics 30*.

CHEN, D., AND PLEMMONS, R. 2007. Nonnegativity constraints in numerical analysis. In *Symposium on the Birth of Numerical Analysis*.

CLUTTERBUCK, S., AND JACOBS, J., 2010. A physically based approach to virtual character deformations. SIGGRAPH 2010 Talks.

CURTIS, S., TAMSTORF, R., AND MANOCHA, D. 2008. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, 61–69.

DE BERG, M., CHEONG, O., VAN KREVELD, M., AND OVERMARS, M. 2008. *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer-Verlag.

DUFF, I. S. 2004. MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Trans. Math. Softw. 30*, 2 (June), 118–144.

GIBSON, S. F., AND MIRTICH, B. 1997. A Survey of Deformable Models in Computer Graphics. Tech. Rep. TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, November.

HARMON, D., AND ZORIN, D. 2013. Subspace integration with local deformations. *ACM Transactions on Graphics 32* (July).

HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. 2009. Asynchronous contact mechanics. *ACM Trans. Graph.*.

HIROTA, G., FISHER, S., STATE, A., LEE, C., AND FUCHS, H. 2001. An implicit finite element method for elastic solids in contact. In *Proceedings of the Fourteenth Conference on Computer Animation*, 136 –254.

HOGG, J., AND SCOTT, J. 2010. An indenite sparse direct solver for multicore machines. Tech. Rep. TR-RAL-2010-011, Rutherford Appleton Laboratory, Chilton, Oxfordshire, UK.

IDELSOHN, S., AND CARDONA, A. 1985. A reduction method for nonlinear structural dynamic analysis. *Computer Methods in Applied Mechanics and Engineering 49*, 253–279.

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

JONES, M. T., PLASSMANN, P. E., AND MCS-P, P. 1995. An improved incomplete cholesky factorization. *ACM Trans. Math. Software 21*, 5–17.

KAVAN, L., AND SORKINE, O. 2012. Elasticity-inspired deformers for character articulation. *ACM Trans. Graph. 31*, 6 (Nov.), 196:1–196:8.

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

KIM, T., AND DELANEY, J. 2013. Subspace fluid re-simulation. *ACM Trans. Graph. 32* (July).

KIM, T., AND JAMES, D. L. 2011. Physics-based character skinning using multi-domain subspace deformations. In *ACM SIGGRAPH/Eurographics Sym. on Computer Animation*, 63–72.

KIM, J., AND POLLARD, N. S. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Trans. Graph. 30*, 5 (Oct.), 121:1–121:19.

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

KURIHARA, T., AND MIYATA, N. 2004. Modeling deformable human hands from medical images. In *ACM SIGGRAPH/Eurographics Sym. on Computer Animation*, 357–366.

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

LIU, L., YIN, K., WANG, B., AND GUO, B. 2013. Simulation and control of skeleton-driven soft body characters. *ACM Trans. Graph. 32*, 6 (Nov.), 215:1–215:8.

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. Graph. 30*, 4 (July), 37:1–37:12.

NEALEN, A., MULLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. In *Eurographics: State of the Art Report*.

O'BRIEN, J. F., SHEN, C., AND GATCHALIAN, C. M. 2002. Synthesizing sounds from rigid-body simulations. In *ACM SIGGRAPH Sym. on Computer Animation*, 175–181.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, 215–222.

SCHVARTZMAN, S. C., GASCÓN, J., AND OTADUY, M. A. 2009. Bounded normal trees for reduced deformations of triangulated surfaces. In *ACM SIGGRAPH/Eurographics Sym. on Computer Animation*, 75–82.

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

SIFAKIS, E., AND BARBIČ, J. 2012. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses*, 20:1–20:50.

STANTON, M., SHENG, Y., WICKE, M., PERAZZI, F., YUEN, A., NARASIMHAN, S., AND TREUILLE, A. 2013. Non-polynomial galerkin projection on deforming meshes. *ACM Trans. Graph. 32* (July).

TANG, M., MANOCHA, D., OTADUY, M. A., AND TONG, R. 2012. Continuous penalty forces. *ACM Trans. Graph. 31*, 4.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *ACM SIGGRAPH Symp. on Computer Animation*, 181–190.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Trans. Graph. 25*, 3 (July), 826–834.

VAILLANT, R., BARTHE, L., GUENNEBAUD, G., CANI, M.-P., ROHMER, D., WYVILL, B., GOURMEL, O., AND PAULIN, M. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*, 4 (July), 125:1–125:12.

VON TYCOWICZ, C., SCHULZ, C., SEIDEL, H.-P., AND HILDEBRANDT, K. 2013. An efficient construction of reduced deformable objects. *ACM Trans. Graph. 32*, 6 (Nov.), 213:1–213:10.

WÄCHTER, A., AND VIGERSKE, S., 2005. Interior point optimizer. https://projects.coin-or.org/Ipopt.

WÄCHTER, A. 2002. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Trans. Graph. 28*, 3 (Aug.), 39.

YANG, Y., XU, W., GUO, X., ZHOU, K., AND GUO, B. 2013. Boundary-aware multi-domain subspace deformation. *IEEE Transactions on Visualization and Computer Graphics*.

ZHU, Y., SIFAKIS, E., TERAN, J., AND BRANDT, A. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29*, 2 (Mar.), 16:1–16:18.