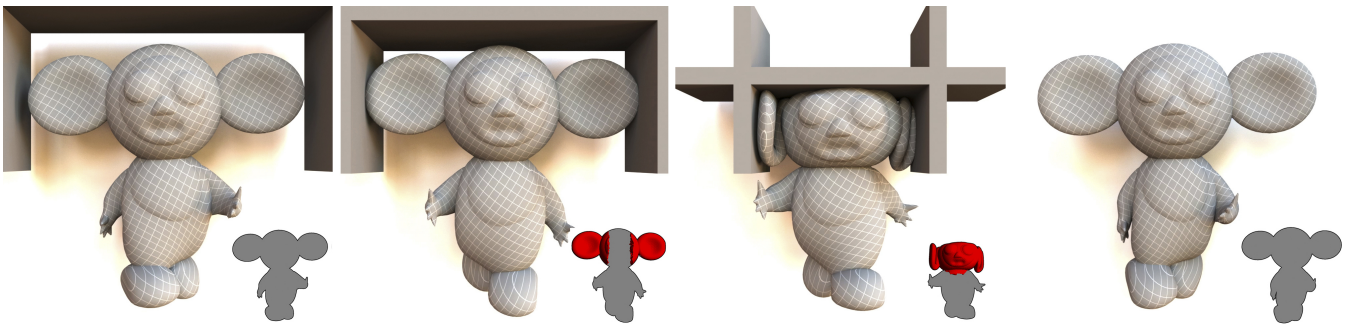
Subspace Condensation: Full Space Adaptivity for Subspace Deformations

Yun Teng*[1,2], Mark Meyer†[2], Tony DeRose‡[2] and Theodore Kim§[1]

[1]University of California, Santa Barbara [2]Pixar Animation Studios



(a) (b) (c) (d)

Figure 1: (a) *The simulation runs at 16 FPS, entirely within the subspace,* **67**$\times$ *faster than a full space simulation over the entire mesh.* **(b)** *Novel wall collisions begin, activating full space tets, shown in red in the inset. The simulation still runs at 2.1 FPS, a* **7.7**$\times$ *speedup.* **(c)** *Collisions produce a deformation far outside the basis, and 49% of the tets are simulated in full space. The step runs at 0.5 FPS; still a* **1.9**$\times$ *speedup.* **(d)** *The collisions are removed, and the* **67**$\times$ *speedup returns.*

Abstract

Subspace deformable body simulations can be very fast, but can behave unrealistically when behaviors outside the prescribed subspace, such as novel external collisions, are encountered. We address this limitation by presenting a fast, flexible new method that allows full space computation to be activated in the neighborhood of novel events while the rest of the body still computes in a subspace. We achieve this using a method we call *subspace condensation*, a variant on the classic *static condensation* precomputation. However, instead of a precomputation, we use the speed of subspace methods to perform the condensation at *every frame*. This approach allows the full space regions to be specified arbitrarily at runtime, and forms a natural two-way coupling with the subspace regions. While condensation is usually only applicable to linear materials, the speed of our technique enables its application to nonlinear materials as well. We show the effectiveness of our approach by applying it to a variety of articulated character scenarios.

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

Keywords: character simulation, subspace integration, static condensation, cubature, collision resolution

*yunteng.cs@cs.ucsb.edu
†mmeyer@pixar.com
‡derose@pixar.com
§kim@mat.ucsb.edu

1 Introduction

Subspace methods, also known as reduced-order, reduced coordinate or model reduction methods, have recently made great strides in accelerating deformable body simulations. In lieu of a full space method, also known as a full-order or full-coordinate method, which simulates every degree of freedom in a mesh, subspace methods instead constrain the motion to a subspace spanned by a compact, but expressive, set of basis vectors. Since r basis vectors are being simulated instead of N vertices, if $r \ll N$, very large speedups can be realized.

An obvious limitation arises when the expressivity of the basis vectors is insufficient, and the true, full space motion of the mesh lies outside the span of the subspace. A straightforward example of this is an external collision, such as a cannonball hitting the mesh in a novel location that was not accounted for when constructing the subspace. In these cases, subspace methods can "lock", producing motions that are both very different from the full space solution and unrealistic in appearance (Fig. 2). A variety of strategies have been developed to account for this situation, including basis enrichment [Harmon and Zorin 2013], adaptive basis construction [Hahn et al. 2014], and falling back to brute-force, full-order computation over the entire mesh [Kim and James 2009].

Figure 2: *Novel collisions cause extreme locking in a subspace-only simulation.*

In this paper, we present a distinctly different approach. We observe that in many cases, particularly when dealing with external collisions, subspace methods only diverge from the full space solution in spatially localized patches. Unfortunately, there is no way to know during the precomputation stage where these patches will be,

and how their locations will change over time. Therefore, we propose an approach that activates full space computation along these patches at run-time, but allows the rest of the model, where the subspace approximation is still valid, to continue computing efficiently in the subspace. We accomplish this using a method we call *subspace condensation*, which is a variation on the widely known *static condensation* method from the domain decomposition literature [Bathe 2007]. While condensation has traditionally been deployed as a precomputation for linear materials, we show that it can be efficiently computed at runtime, even for non-linear materials, by leveraging the reduced dimensionality of the subspace. The method does not require any constraint mechanisms such as spring forces [Kim and James 2011] or Lagrange multipliers [Yang et al. 2013], to couple the subspace and full space regions.

Our method allows subspace methods to be used in cases where they previously would not have been considered. Even if it is known in advance that novel behaviors will arise, we provide a mechanism that allows the simulation to only "pay for" the novel components in each frame, while other, more familiar behaviors are solved efficiently. Our method gracefully degrades, so in the worst case scenario, it merely falls back to a full space simulation over the entire mesh. We demonstrate the effectiveness of our algorithm on a variety of character animation examples. Our main contributions are:

- *Subspace condensation*, a new method that combines the generality of full space deformations with the speed of subspace computations.
- The main bottleneck of condensation methods is a large matrix inverse. We design a solver that sidesteps this problem using subspace coordinates, but still maintains a two-way coupling between the full space and subspace regions.
- Condensation is usually only applicable to linear materials, but the speed of our method allows it to be applied to non-linear materials as well.
- We demonstrate our algorithm on a physics-based skinning application. To this end, we propose several *oracles* that detect the regions where full space computation is needed and where the subspace approximation will suffice, and dynamically partitions the mesh into these regions at every frame.
- By exploiting the forces along the boundary of the full space regions, we show that an efficient, cubature-based method [An et al. 2008; von Tycowicz et al. 2013] can be obtained for evaluating the forces inside the subspace regions.

2 Related Work

Simulating deformable objects is a well-studied subject in computer graphics, and excellent articles exist that discuss developments up through the 1990s [Gibson and Mirtich 1997], the 2000s [Nealen et al. 2005] and approaching the present day [Sifakis and Barbič 2012]. We are particularly interested in *subspace* methods, also known as reduced-order, reduced coordinate, or model reduction methods, for accelerating these simulations. These methods have been known for some time in computer graphics [Pentland and Williams 1989; Hauser et al. 2003], and have seen recent interest due to successes in incorporating non-linear phenomena [Barbič and James 2005; An et al. 2008; Li et al. 2014].

Subspace methods replace the N nodal degrees of freedom in a deformable body with a subspace of r basis vectors. It has been widely observed that when $r \ll N$, a broad span of relevant deformations can still be efficiently captured. Inevitably, deformations that are not well-captured by the subspace arise, particularly when novel loadings are applied that were not accounted for during subspace construction.

A variety of strategies have been devised to address this limitation. The basis vectors usually have global support, so domain decomposition techniques, also known as substructuring techniques, have been used to localize their influence and reduce the likelihood that a novel local deformation will trigger a non-physical, global artifact [Barbič and Zhao 2011; Kim and James 2011; Yang et al. 2013]. To avoid confusion with other decompositions of the simulation domain that we use in this paper, we will refer to these more specifically as "skeletal decomposition" techniques.

Many enrichment techniques have also been proposed, such as the use of approximate, analytic Boussinesq solutions [Harmon and Zorin 2013], or the construction of a large database that is used to build a custom subspace model at every frame [Hahn et al. 2014; Xu et al. 2014b; Teng et al. 2014]. While these methods are successful at making subspace methods more general, they can still be defeated by novel deformations that are not well-captured by the Boussinesq approximation, or not present in the database. Our method complements these existing ones; it can be activated at the moment that they fail.

Our method is based on *static condensation*, an algorithm that has been known in structural mechanics and civil engineering for some time [Guyan 1965; Irons 1965; Wilson 1974] and was originally developed for static vibration analysis, i.e. the eigenmodes of a structure. For this reason, the procedure is also sometimes referred to as "eigenvalue economization" [Leung 1978]. Practitioners are naturally also interested in dynamics, so *dynamic condensation* was developed to take inertial effects into account [Leung 1978; Paz 1989]. In graphics, we are by no means the first researchers to leverage this technique, as it was used successfully by Bro-Nielsen and Cotin [1996] for real-time surgery. More recently, it was leveraged successfully in the context of physically-based skinning [Gao et al. 2014], where a very nice connection to the Steklov-Poincaré operator was also drawn, and was applied in a novel material optimization context by Xu and colleagues [2014a]. Related techniques exist that also explore the possibility of "surface-only" volumetric simulations [James and Pai 2003].

Our work differs from the two most related graphics works [Bro-Nielsen and Cotin 1996; Gao et al. 2014] in two key ways. First, due to the presence of an expensive matrix inverse, they either perform the condensation as a pre-process or build a special material model whose inverse is easier to compute. We show that by leveraging fast subspace inverses, condensation can be performed quickly and dynamically at run-time for an arbitrary material.

Second, they assume that the surface degrees of freedom (DOFs) are the most important ones, and use condensation to project away the interior DOFs. Our method allows *any* subset of nodes to be designated as important, allows these designates to be changed at every frame, and can quickly project away the complexity of their complement. This distinction is significant, because in the case of local, non-trivial contacts, DOFs on the interior of the mesh can play a significant role in the appearance of the final deformation, and should not always be be projected away. Our general approach allows any subset of interior DOFs to be simulated as necessary, and the surface-only variant becomes a special case. We elaborate on these distinctions in the next section.

3 Combining Full Space and Subspace Simulations

Throughout this paper, we will use the following notation. Bold lowercase symbols denote a vector, e.g. $\mathbf{f}$, while bold uppercase de-

notes a matrix, e.g. $\mathbf{K}$. Unbolded symbols represent scalars. The reserved symbol N represents the full-order rank of a mesh, i.e. the number of unconstrained vertices in a tetrahedral mesh, and the symbol r denotes the subspace rank. The matrix $\mathbf{U} \in \mathbb{R}^{3N \times r}$ then represents the subspace basis that efficient operations are performed in. Subspace quantities are denoted with a tilde, such as $\widetilde{\mathbf{f}} = \mathbf{U}^T\mathbf{f}$, which is in $\mathbb{R}^r$, and $\widetilde{\mathbf{K}} = \mathbf{U}^T\mathbf{K}\mathbf{U}$, which is in $\mathbb{R}^{r \times r}$.

3.1 Static Condensation

We will first review the basics of static condensation before describing our novel variant. For consistency, we adhere to the notation of Bro-Nielsen and Cotin [1996] where possible. Let $\mathbf{K}\mathbf{u} = \mathbf{f}$ be the linearized, quasistatic system that models a solid object. Here, $\mathbf{K} \in \mathbb{R}^{3N \times 3N}$, $\mathbf{u} \in \mathbb{R}^{3N}$, and $\mathbf{f} \in \mathbb{R}^{3N}$. If we reorder the vertices so that the external surface vertices (V_e) come before the internal vertices (V_i), we can rewrite the system as a block structure:

$$\begin{bmatrix} \mathbf{K}_{ee} & \mathbf{K}_{ei} \\ \mathbf{K}_{ie} & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{u}_e \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}_e \\ \mathbf{f}_i \end{bmatrix} \tag{1}$$

Here, e indicates external (i.e. surface) vertices and i indicates internal vertices, where $|V_e| + |V_i| = N$. The $\mathbf{K}_{ie} \in \mathbb{R}^{3|V_i| \times 3|V_e|}$ and $\mathbf{K}_{ei} \in \mathbb{R}^{3|V_e| \times 3|V_i|}$ blocks represent the couplings between the two sets. By using 2×2 block Gaussian elimination, we obtain a system that only involves the surface vertices,

$$\mathbf{K}^*_{ee}\mathbf{u}_e = \mathbf{f}^*_e, \tag{2}$$

where

$$\mathbf{K}^*_{ee} = \mathbf{K}_{ee} - \mathbf{K}_{ei}\mathbf{K}_{ii}^{-1}\mathbf{K}_{ie} \tag{3}$$

$$\mathbf{f}^*_e = \mathbf{f}_e - \mathbf{K}_{ei}\mathbf{K}_{ii}^{-1}\mathbf{f}_i. \tag{4}$$

Eqn. 3 is the well-known Schur complement, a widely used expression in domain decomposition, and Eqns. 3 and 4 together form its standard application. After solving for $\mathbf{u}_e$, if $\mathbf{u}_i$ is desired, it can still be retreived via

$$\mathbf{u}_i = \mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{K}_{ie}\mathbf{u}_e). \tag{5}$$

However, if only the external surface positions $\mathbf{u}_e$ are needed, the degrees of freedom of the internal vertices have been condensed away. In the case of linear materials, both Eqn. 3 and the $\mathbf{K}_{ei}\mathbf{K}_{ii}^{-1}$ term in $\mathbf{f}^*_e$ can be precomputed and reused at runtime [Bro-Nielsen and Cotin 1996]. The runtime cost is then drastically reduced, as the inverse at runtime now depends on $|V_e|$, the number of surface vertices, not N, the number of total vertices. Adding dynamics is then a straightforward application of the same block reordering to the mass and damping matrices, $\mathbf{M}$ and $\mathbf{C}$ (see §2.4.2 in [Bro-Nielsen and Cotin 1996]).

Discussion: Static condensation works best in the context of linear materials. In the non-linear case, $\mathbf{K}$ is no longer constant and becomes $\mathbf{K}(\mathbf{u})$, and repeatedly computing the $\mathbf{K}_{ii}(\mathbf{u})^{-1}$ in Eqs. 3 and 4 can be prohibitively expensive. Some progress has been made on this limitation, as Gao and colleagues [2014] recently formulated an *ex-rotated* (extrinsically rotated) material model that is specifically tailored to efficiently approximate $\mathbf{K}_{ii}(\mathbf{u})^{-1}$. We instead take this technique in a different direction. Broadly, the vertices can be partitioned arbitrarily for any purpose, not just according to the internal and external vertices. The question we answer in the affirmative is: is it possible to partition the vertices so that a select few are simulated in the full space, while the remaining are simulated in a subspace?

Partition Oracle: The issue of how the vertices are divided into full space and subspace regions is a separate question that we delegate to an external *partition oracle*. We will propose several oracles in §4.2, but for now will put this issue aside and describe a generic condensation technique that is not dependent on the specifics of any particular oracle.

3.2 Subspace Condensation

The main bottleneck of static condensation is computing the inverse, $\mathbf{K}_{ii}^{-1}(\mathbf{u}) \in \mathbb{R}^{3|V_i| \times 3|V_i|}$. Subspace methods excel at quickly inverting compact approximations to these kinds of matrices, $\widetilde{\mathbf{K}}_{ii}^{-1}(\widetilde{\mathbf{u}}) \in \mathbb{R}^{r \times r}$. Their applicability looks promising, and would enable static condensation for non-linear materials, but several non-trivial issues must be addressed to make the algorithm practical.

We define *full vertices* as those undergoing full space simulation, and denote their set as V_f. The rest are referred to as *subspace vertices* and denoted as V_s. Again, we assume that some external partition oracle has provided these labels. Let us consider quasistatic, non-linear deformations without any external forces. Assuming that V_f is not empty, we reorder the vertices and write the system $\mathbf{K}\mathbf{u} = \mathbf{f}$ as

$$\begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fs} \\ \mathbf{K}_{sf} & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_f \\ \mathbf{f}_s \end{bmatrix}, \tag{6}$$

which can then be repeatedly solved iteratively inside a Newton solver. Here we have abbreviated the non-linear terms $\mathbf{K}(\mathbf{u}) = \mathbf{K}$, $\mathbf{K}_{ff}(\mathbf{u}) = \mathbf{K}_{ff}$, and so on, for brevity. The dimensions are then: $\mathbf{u}_f, \mathbf{f}_f \in \mathbb{R}^{3|V_f|}$, $\mathbf{u}_s, \mathbf{f}_s \in \mathbb{R}^{3|V_s|}$, $\mathbf{K}_{ff} \in \mathbb{R}^{3|V_f| \times 3|V_f|}$, $\mathbf{K}_{ss} \in \mathbb{R}^{3|V_s| \times 3|V_s|}$, $\mathbf{K}_{fs} = \mathbf{K}_{sf}^T \in \mathbb{R}^{3|V_f| \times 3|V_s|}$. We now want to solve for $\mathbf{u}_f$ and $\mathbf{u}_s$, and applying the condensation technique requires the inversion of $\mathbf{K}_{ss}$, much like in Eqns. 3 and 4.

A Matrix Formulation: Fast subspace inverses can be directly, but naïvely, applied to this problem. For example, an analog to Eqn. 4,

$$\mathbf{f}^*_f = \mathbf{f}_f - \mathbf{K}_{fs}\mathbf{K}_{ss}^{-1}\mathbf{f}_s, \tag{7}$$

can incorporate the subspace matrix $\widetilde{\mathbf{K}}_{ss}^{-1}(\widetilde{\mathbf{u}})$ thusly,

$$\mathbf{f}^*_f \approx \mathbf{f}_f - \mathbf{K}_{fs}\left(\mathbf{U}_s\widetilde{\mathbf{K}}_{ss}^{-1}(\widetilde{\mathbf{u}})\mathbf{U}_s^T\right)\mathbf{f}_s, \tag{8}$$

where $\mathbf{U}_s \in \mathbb{R}^{3|V_s| \times r}$ is a basis matrix composed of the rows of $\mathbf{U}$ that correspond to the vertices in V_s. Only a small matrix now needs to be inverted, $\widetilde{\mathbf{K}}_{ss}^{-1}(\widetilde{\mathbf{u}}) \in \mathbb{R}^{r \times r}$, which is quickly done in the subspace, the result is expanded into a larger matrix $\mathbf{U}_s\widetilde{\mathbf{K}}_{ss}^{-1}(\widetilde{\mathbf{u}})\mathbf{U}_s^T \in \mathbb{R}^{3|V_s| \times 3|V_s|}$, and used to compute $\mathbf{f}^*_f \in \mathbb{R}^{3|V_s|}$. An analogous method can be used for Eqn. 3.

Unfortunately, this formulation is highly inaccurate. The $\mathbf{K}_{fs}\mathbf{K}_{ss}^{-1}\mathbf{f}_s \in \mathbb{R}^{3|V_f|}$ term in Eqn. 7 as a corrective force vector to $\mathbf{f}_f$, so it is reasonable to expect that the vector $\mathbf{K}_{ss}^{-1}\mathbf{f}_s \in \mathbb{R}^{3|V_s|}$ lies in the column span of $\mathbf{U}_s \in \mathbb{R}^{3|V_s| \times r}$. That corrective force, or something similar, is exactly what was input into the SVD that constructed the subspace $\mathbf{U}_s$.

However, the $\mathbf{U}_s\widetilde{\mathbf{K}}_{ss}^{-1}(\widetilde{\mathbf{u}})\mathbf{U}_s^T$ term in in Eqn. 8 is an expanded version of $\mathbf{K}_{ss}^{-1}$, the Jacobian of $\mathbf{f}_s$. While $\mathbf{U}_s$ may span the subspace of important $\mathbf{f}_s$ values, there is no reason to believe that it also spans its Jacobian. If this were true, it would imply that $\mathbf{K}_{ss}^{-1}$ has low rank, and that $\mathbf{U}_s$ spans its dominant eigenvectors. Simple numerical experiments verify that neither of these assumptions are true; if $\mathbf{K}_{ss}$ were not full rank, its inverse would not exist.

Instead, it is clear that $\widetilde{\mathbf{K}}_{ss}^{-1}(\widetilde{\mathbf{u}})$ should not be expanded, as it is only a meaningful Jacobian of the reduced force $\widetilde{\mathbf{f}}_s$. Therefore, we need

to structure our algorithm so that not only the inverse is computed quickly, but that the result is carried forward in the subspace until a reduced version of the entire corrective force, $\mathbf{K}_{ss}^{-1}\mathbf{f}_s$, is obtained.

A Vector Formulation: We instead examine Eqn. 6 by expanding it from its block form into

$$\mathbf{K}_{ff}\mathbf{u}_f + \mathbf{K}_{fs}\mathbf{u}_s = \mathbf{f}_f$$
$$\mathbf{K}_{sf}\mathbf{u}_f + \mathbf{K}_{ss}\mathbf{u}_s = \mathbf{f}_s.$$

If we project the entire second equation using $\mathbf{U}_s^T$, and perform the substitution $\mathbf{u}_s \approx \mathbf{U}_s\widetilde{\mathbf{u}}_s$ on both equations, we obtain

$$\mathbf{K}_{ff}\mathbf{u}_f + \mathbf{K}_{fs}\mathbf{U}_s\widetilde{\mathbf{u}}_s = \mathbf{f}_f \tag{9}$$
$$\mathbf{U}_s^T\mathbf{K}_{sf}\mathbf{u}_f + \widetilde{\mathbf{K}}_{ss}\widetilde{\mathbf{u}}_s = \widetilde{\mathbf{f}}_s. \tag{10}$$

The subspace regions now communicate with the full space through $\widetilde{\mathbf{u}}_s$, namely the $\mathbf{K}_{fs}\mathbf{U}_s\widetilde{\mathbf{u}}_s$ product, not through a rank-deficient expansion of the $\widetilde{\mathbf{K}}_{ss}^{-1}$ matrix. The system can be returned to block form:

$$\begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fs}\mathbf{U}_s \\ \mathbf{U}_s^T\mathbf{K}_{sf} & \widetilde{\mathbf{K}}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \widetilde{\mathbf{u}}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_f \\ \widetilde{\mathbf{f}}_s \end{bmatrix}. \tag{11}$$

Solving this new system (Eqn. 11) for $\mathbf{u}_f$ and $\widetilde{\mathbf{u}}_s$ yields exactly the subspace-to-full space coupling that we seek. We perform all computations in the subspace until a subspace displacement vector $\widetilde{\mathbf{u}}_s$ is obtained. Expanding $\widetilde{\mathbf{u}}_s$ using $\mathbf{U}_s$ then uses the basis matrix for its intended purpose. In order to solve the system efficiently, one additional component is needed, which we will now describe.

Efficient Force Evaluation: It is not immediately obvious how to efficiently compute $\widetilde{\mathbf{f}}_s$, the internal forces on the subspace vertices. A brute-force method would be to compute and project the full space force $\mathbf{f}_s$ [Krysl et al. 2001]. However, this would make its evaluation $O(N)$, a complexity that we explicitly want to avoid.

One approach would be to quickly approximate the force over *all* vertices, $\widetilde{\mathbf{f}} \in \mathbb{R}^r$, using an existing subspace method [Barbič and James 2005; An et al. 2008], and subtract off the forces from the full space vertices, $\mathbf{f}_f$, which should not participate in the subspace solve. We can express this as $\widetilde{\mathbf{f}}_s \approx \widetilde{\mathbf{f}} - \mathbf{U}_f^T\mathbf{f}_f$, where $\mathbf{U}_f \in \mathbb{R}^{3|V_f|\times r}$ denotes the rows of $\mathbf{U}$ that correspond to the full vertices, V_f. However, since the full space force can contain components that are not well-captured by $\mathbf{U}_f$, the projection produces unusable results.

Fortunately, we are able to devise an efficient, alternative, cubature-based [An et al. 2008; von Tycowicz et al. 2013] method. First, we only evaluate the cubature tets that lie inside the set of subspace tets, T_s. Second, we project the forces exerted by the tets on the *boundary* between the full space and subspace region, which are already available from evaluating $\mathbf{f}_f$. Unlike in the previous case, these boundary forces are very likely to be well-captured by the basis, as they have already been partially constrained to the subspace during previous timesteps. The remainder of the full space region (Fig. 3, red), whose out-of-basis projections will likely just produce locking artifacts, is correctly ignored. More formally:

$$\widetilde{\mathbf{f}}_s \approx \sum_{j\in V_b} \mathbf{U}_j^T\mathbf{f}_j + \sum_{i\in T_c} \delta\cdot w_i\cdot \mathbf{U}_i^T\mathbf{f}_i(\widetilde{\mathbf{u}}) \begin{cases} \delta = 1 & \text{if } i \in T_s \\ \delta = 0 & \text{if } i \in T_f \end{cases}. \tag{12}$$

In the first summation, V_b is the set of subspace vertices on the boundary between the full space and subspace regions, $\mathbf{f}_j$ is the internal force on the jth boundary vertex exerted by its adjacent boundary tets, and $\mathbf{U}_j$ is the rows of $\mathbf{U}$ that correspond to that vertex. The second summation is the cubature approximation [An et al. 2008], where T_c is the set of cubature tets, w_i denotes the weight on cubature tet i, $\mathbf{U}_i$ the rows of $\mathbf{U}$ that correspond to the vertices in tet i, and $\mathbf{f}_i$ is the internal force function evaluated at tet i. Fig. 3 shows a 2D illustration of this process. The total time to evaluate Eqn. 12 is $O(|T_c| + |V_b|)$, which is proportional to the complexity of the subspace and the currently activated full vertices. The stiffness matrix on the subspace vertices $\widetilde{\mathbf{K}}_{ss}$ can be computed in the same manner. Any $O(N)$ dependency on the full complexity of the model has been removed.

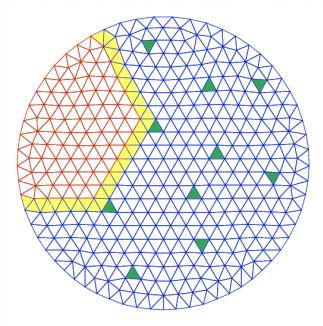


Figure 3: *A 2D illustration of Eqn. 12. The $\widetilde{\mathbf{f}}_s$ vector is computed by projecting the force exerted by boundary tets (yellow) onto the subspace region (blue) and adding the weighted forces from the cubature tets (green) that reside completely in the subspace region (blue).*

Solving the System: While it is be possible to apply the traditional Schur complement to Eqn. 11, it is both undesirable and unnecessary. We always want to solve for $\widetilde{\mathbf{u}}_s$, as it is needed to update the surface positions in the subspace region. Unlike the classic static condensation case, $\widetilde{\mathbf{u}}_s$ is very small, so computing it in addition to $\mathbf{u}_f$ does not add prohibitive complexity. This also allows us to trivially apply the method to non-linear materials, as we can simply solve a few Newton iterations of $\widetilde{\mathbf{u}}_s$.

We initially attempted to solve Eqn. 11 using a sparse direct solver, but matrix assembly was taking up 25∼30% of the entire simulation time, so we switched to preconditioned conjugate gradient (PCG) instead. We use a Jacobi preconditioner for the upper diagonal block, and precondition the lower diagonal block with its explicit inverse ($\widetilde{\mathbf{K}}_{ss}^{-1}$). In our examples the solver typically converges to a relative error of 0.01 in fewer than 100 iterations. Using a more sophisticated preconditioner such as Incomplete Cholesky (IC) is difficult, because each factorization sees limited re-use before V_f changes and it has to be recomputed. Ideally a multigrid method would be applied to this system, but we leave this problem as future work.

The entire solution procedure now depends on the rank r of the subspace, and $|V_f|$, the number of active full space vertices. The $\mathbf{K}_{fs}\mathbf{U}_s$ product and its transpose in Eqn. 11 at first appears to be $O(N \times r)$, but $\mathbf{K}_{fs}$ is very sparse, and contains $O(|V_f|)$ non-zero entries. Since $\mathbf{K}_{fs}$ encodes the coupling between $O(|V_f|)$ vertices and the rest of the mesh, this sparsity is to be expected. The overall solver is outlined in Algorithm 1, and it is clear that no $O(N)$ computation is needed at any stage.

4 Physics-Based Skinning

Our simulation technique is general enough to be applied to any subspace deformable body simulation. As a proof-of-concept, we have applied it to the problem of novel external collisions in physics-based skinning. We will describe the features of the skinning technique here, while noting that the subspace condensation technique does not fundamentally rely on any of them.

Algorithm 1 Integration using subspace condensation

construct active full space region V_f
for $i := 1$ **to** n **do** ▷ n = # of Newton iterations
  if $V_f \neq \emptyset$ **then**
    initialize $\widetilde{\mathbf{u}}_s = \mathbf{0}$
    compute $\mathbf{f}_f, \mathbf{K}_{ff}, \mathbf{U}_s^T\mathbf{K}_{sf}$ $\left(= (\mathbf{K}_{fs}\mathbf{U}_s)^T\right)$
    compute $\widetilde{\mathbf{f}}_s$ and $\widetilde{\mathbf{K}}_{ss}$ using cubature (Eqn. 12)
    solve Eqn. 11 for $\mathbf{u}_f$ and $\widetilde{\mathbf{u}}_s$.
    update full space region using $\mathbf{u}_f$
    update subspace region using $\widetilde{\mathbf{u}}_s$
  else
    perform subspace-only Newton step over entire mesh
  end if
end for

4.1 Basis Construction

We elected to use a skinning correction basis, similar to that used by EigenSkin [Kry et al. 2002] and the kinematic correction employed by Hahn et al. [2014]. The basis is constructed by first subtracting off the skinning transformation from the simulation result, and then performing PCA. The subspace then serves as a physics-based corrective to the purely "kinematic" skinning model. Keeping the notation similar to the latter paper where possible, φ denotes a skinning function, and the final deformed, world-space position $\mathbf{x}$ for a single vertex m is

$$\begin{aligned}\mathbf{x}^m &= \overline{\mathbf{x}}^m + \mathbf{u}^m \\ &= \varphi(\mathbf{p}, \overline{\mathbf{x}}^m + \overline{\mathbf{u}}^m) = \mathbf{R}(\overline{\mathbf{x}}^m + \overline{\mathbf{u}}^m) + \mathbf{t}.\end{aligned}$$

Here, $\mathbf{p}$ is the current skeleton configuration, e.g. as expressed by joint angles, $\overline{\mathbf{x}}^m$ denotes the rest pose, $\mathbf{u}^m$ is the world-space displacement, and $\overline{\mathbf{u}}^m$ is the same displacement prior to the skinning transformation. $\mathbf{R}$ and $\mathbf{t}$ together represent the affine transformation determined by the skinning function. In order to obtain this transform, we used dual-quaternion skinning [Kavan et al. 2007] and volumetric heat diffusion [Baran and Popović 2007] to propagate the weights throughout the tetrahedral mesh. One of the nice properties of dual-quaternion skinning is that $\mathbf{R}$ is guaranteed to be a pure rotation matrix. This will later be used to simplify the skeletal decomposition forces.

In order to build our basis $\mathbf{U}$, we first obtained samples of the global displacement vector $\mathbf{u}$ using full space simulations over the entire mesh. We then inverted the skinning transform φ^{-1} to pull each $\mathbf{u}$ back to its untransformed version $\overline{\mathbf{u}}$. Our subspace basis $\mathbf{U}$ is then constructed by performing a truncated PCA over these samples of $\overline{\mathbf{u}}$. The world-space, full space position is then computed as:

$$\mathbf{x} = \varphi(\mathbf{p}, \overline{\mathbf{x}} + \mathbf{U}\widetilde{\mathbf{u}}). \tag{13}$$

As observed in previous work, this basis is significantly more flexible than the one obtained by performing PCA directly over $\mathbf{u}$. The skinning transforms have been factored out, so the correctives that remain can be applied over a wide range of scenarios.

Skeletal Decomposition: The role of the skinning transform φ in the basis could potentially complicate the use of skeletal decomposition techniques, such as the one from Kim and James [2011]. In that work, penalty springs were inserted between bone-centered domains to ensure their compatibility. A 3rd-order "fast sandwich transform" (FST) tensor had to be introduced to efficiently incorporate each domain's local rotation into the subspace computation.

In our work, we found that the choice of a skinning correction basis *removes* the need for any special transforms. To see why this is so, we examine the penalty spring energy E^m between two domains, i and j. Let $\mathbf{v}^m$ be an interface vertex between these two domains with respective positions of $\mathbf{x}_i^m$ and $\mathbf{x}_j^m$. The spring energy is then written as:

$$E^m = \frac{1}{2}k\,(\mathbf{x}_i^m - \mathbf{x}_j^m)^T(\mathbf{x}_i^m - \mathbf{x}_j^m) \tag{14}$$

$$\begin{aligned}&= \frac{1}{2}k\,[\varphi(\mathbf{p}, \overline{\mathbf{x}}^m + \overline{\mathbf{u}}_i^m) - \varphi(\mathbf{p}, \overline{\mathbf{x}}^m + \overline{\mathbf{u}}_j^m)]^T \\ &\qquad [\varphi(\mathbf{p}, \overline{\mathbf{x}}^m + \overline{\mathbf{u}}_i^m) - \varphi(\mathbf{p}, \overline{\mathbf{x}}^m + \overline{\mathbf{u}}_j^m)]\end{aligned} \tag{15}$$

$$\begin{aligned}&= \frac{1}{2}k\,[\mathbf{R}_j^m(\mathbf{U}_i^m\widetilde{\mathbf{u}}_i - \mathbf{U}_j^m\widetilde{\mathbf{u}}_j)]^T \\ &\qquad [\mathbf{R}_i^m(\mathbf{U}_i^m\widetilde{\mathbf{u}}_i - \mathbf{U}_j^m\widetilde{\mathbf{u}}_j)].\end{aligned} \tag{16}$$

Here k is the spring constant, $\mathbf{U}^m \in \mathbb{R}^{3\times r}$ is the basis for vertex $\mathbf{v}^m$, and $\mathbf{R}_i^m$ and $\mathbf{R}_j^m$ are the rotations for the vertex in partitions i and j. The solver then requires the gradient (i.e. the spring force) and the Hessian of E^m with respect to $\widetilde{\mathbf{u}}_i$:

$$\frac{\partial E^m}{\partial \widetilde{\mathbf{u}}_i} = k(\mathbf{U}_i^m)^T(\mathbf{R}_j^m)^T\mathbf{R}_i^m(\mathbf{U}_i^m\widetilde{\mathbf{u}}_i - \mathbf{U}_j^m\widetilde{\mathbf{u}}_j) \tag{17}$$

$$\frac{\partial^2 E^m}{\partial \widetilde{\mathbf{u}}_i^2} = k(\mathbf{U}_i^m)^T(\mathbf{R}_j^m)^T\mathbf{R}_i^m\mathbf{U}_i^m. \tag{18}$$

In the previous work [Kim and James 2011], the composition of rotations, $(\mathbf{R}_i^m)^T\mathbf{R}_j^m$, was then fed into an FST tensor. However, under the current basis, the skinning guarantees the two rotations will always be the same, $\mathbf{R}_i^m = \mathbf{R}_j^m$, so the composition $(\mathbf{R}_i^m)^T\mathbf{R}_j^m$ is always the identity matrix. The gradient then becomes,

$$\frac{\partial E^m}{\partial \widetilde{\mathbf{u}}_i} = k((\mathbf{U}_i^m)^T\mathbf{U}_i^m\widetilde{\mathbf{u}}_i - (\mathbf{U}_i^m)^T\mathbf{U}_j^m\widetilde{\mathbf{u}}_j), \tag{19}$$

where everything aside from $\widetilde{\mathbf{u}}_i$ and $\widetilde{\mathbf{u}}_j$ can be precomputed, and the Hessian resolves to a constant matrix,

$$\frac{\partial^2 E^m}{\partial \widetilde{\mathbf{u}}_i^2} = k(\mathbf{U}_i^m)^T\mathbf{U}_i^m. \tag{20}$$

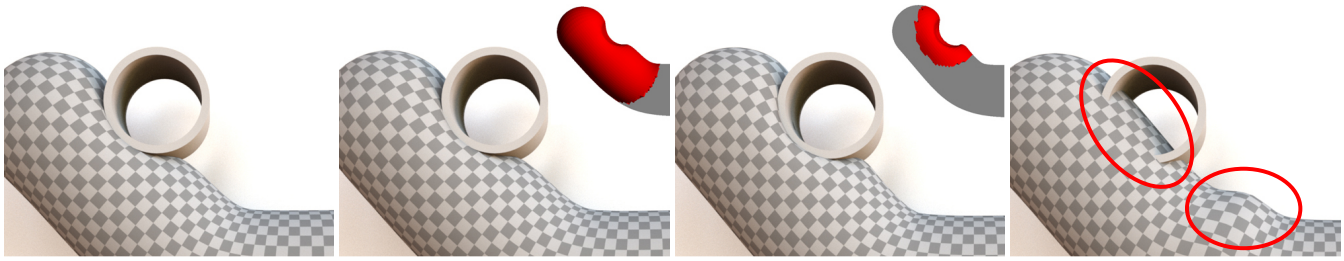
Other gradient and Hessian terms can be deduced similarly.

4.2 Contact and Dynamics Oracles

We applied subspace condensation to the problem of contact handling in both quasistatic and dynamic simulations, and used the penalty-based collision force model from McAdams et al. [2011] for both external and self-collisions.

Contact Oracle: As mentioned in §3.1, some form of *partition oracle* is needed to label the full space and subspace regions. When contacts are the main source of novel deformations, it is natural to build an oracle that is based on collisions. We labelled the vertices that are in collision as belonging to the full space region, and additionally applied a simple, distance-based criterion. Specifically, we conducted a breadth-first search that started from each colliding vertex and terminated when the vertices within an influence radius ρ of the starting vertex had been included. All of the vertices encountered during this search were added to the full space region. Consequently, all collision-related force and Hessian terms only exist in the $\mathbf{f}_f$ and $\mathbf{K}_{ff}$ terms in Eqn. 11. The radius ρ provided a speed-quality tradeoff that will be discussed in detail in §5.

When the mesh is recovering from a complex contact, we found that it is inadvisable to deactivate the full space region as soon as no collisions are detected. The subspace basis has no knowledge of the deformation created by the collision, and can have trouble



(a) Full space solution (b) Subspace Condensation ($\rho = .25$) (c) Subspace Condensation ($\rho = .08$) (d) Subspace-only solution

Figure 4: *For (b) and (c), activated full space region is shown in red in the inset. While the deformation in (c) is less pronounced than the ground truth, the contact is still resolved. (d) Subspace-only simulation cannot resolve the contact. Locking artifacts are circled in red.*

generating the detailed, localized, restoring force necessary to untangle a configuration, e.g. a fold that formed in the palm of a hand. Therefore, the oracle was modified to also include the vertices discovered by the breadth-first search from the previous frame. After two frames, the search results time out, which allow the full space region to shrink.

Dynamics Oracle: Our subspace condensation approach can be applied to dynamics simulation by adding another modification to the oracle. Even if a body is no longer in collision, accelerations caused by the contact forces can lead to interesting deformations that are not captured by the subspace basis. Therefore, when simulating dynamics, the oracle only folds a full space region back into the subspace if both the average velocity and acceleration for the full vertices are below a certain threshold. We found this simple strategy to be effective, though somewhat conservative.

5 Results

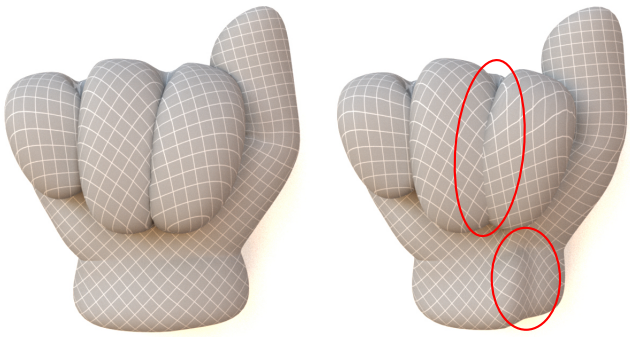
We use Newton's method as our non-linear solver and implicit Euler for our time discretization. Our full rank and subspace condensation solvers use Jacobi-preconditioned Conjugate Gradients for solving the linearized systems (Table 1). Dense linear algebra routines, such as the direct subspace inverse, use the Eigen [Guennebaud et al. 2010] library. Except for Sullivan (Fig. 8), all simulations were run on a relatively modest 8-core, 2.4 Ghz MacBook Pro with 8 GB of RAM using 8 threads. OpenMP was used to parallelize the force and stiffness computation across different skeletal domains, as well as collision detection in both the subspace and full space simulations. All of our results used the co-rotational material from McAdams et al. [2011]. A performance summary of all the examples is shown in Table 2. The reported influence radii all assume that the mesh has been normalized to a unit cube.

Example	Influence Radius	Max. PCG Iterations	Avg. PCG Iterations
Capsule	0.25	126	65
	0.08	24	15
Hand	0.046	86	21
Cheb	0.39	76	29
Sullivan	0.03 (walk)	41	17
	0.048 (belly pat)	53	17

Table 1: *PCG iterations for each example. All examples converged to a relative error of 0.01. Even with a simple Jacobi preconditioner, the solver usually converges quickly.*

Capsule: As an initial test case, we simulated a capsule containing 29,343 vertices and 160,960 tetrahedra. We rigged the capsule with 2 bones and trained the subspace using 12 bending poses, producing a basis with rank 11. Fig. 4(a) shows the full space simulation of the capsule bending and colliding against a pipe. The subspace-only simulation cannot resolve the novel contact deformation, so we observe severe locking (Fig. 4(d)).

Our subspace condensation method easily handles this situation by adaptively activating full space vertices near the contact region. With a small influence radius ($\rho = 0.08$) we obtain a useful approximation to the full solution with an average speedup of $76\times$ (Fig. 4(c)). Increasing the ρ to 0.25 refines the approximation further (Fig. 4(b)) and still maintains an average speedup of nearly an order of magnitude ($9.4\times$). In the worst case, 50.6% of the vertices are simulated in full space, but we still see a $3.5\times$ speedup.



(a) Subspace Condensation (b) Subspace-only solution

Figure 5: *Comparison between our approach with subspace-only simulation on a highly novel contact configuration. The subspace-only solution produces significant locking artifacts, circled in red.*

Hand: The hand mesh in Figs. 5 and 9 consists of 95,746 vertices and 458,071 tetrahedra, and is rigged with 10 bones. The subspace was constructed using a simple, sparse sampling of 53 snapshots obtained by actuating each bone in isolation. The basis has no knowledge of multiple bones moving in tandem. The total rank of the subspace bases, summed over all the skeletal domains, is 156. Self-collisions were taken into account during training, so we used these samples to compute self-collision cubature (SCC) [Teng et al. 2014] in order to quickly compute similar joint collisions at runtime. The SCC was quickly computed using the NN-HTP algorithm [von Tycowicz et al. 2013].

We then put the hand through a quasistatic, calisthenic exercise regimen. The complete sequence is shown in the supplement video. The subspace-only simulation easily handled different combinations of individual joint motions as long as there was no novel contact. However, it immediately failed when novel collisions occurred

(Fig. 5(b)). In contrast, our approach was able to resolve arbitrary contact by adaptively activating full space vertices near the contact areas (Fig. 5(a)). Collisions seen during training were handled using self-collision cubature unless they overlapped with the full space region. In this case, the entire region was treated as a novel contact. On average, our simulation ran at 0.28 s/frame, accelerating the full space simulation by 48×. Even in the worst case, the most novel collision-induced deformation was computed 8.1× faster than a full space simulation over the entire mesh. Fig. 9 shows the time per frame, as well as the percentage of vertices being simulated in full space, for a fist clenching sequence. The simulation time is clearly proportional to the size of the full space region.

To test for convergence of our approach, we compared the simulation errors of using different influence radii against the full space simulation solution for a subsequence of the hand motion (Fig. 7). We used relative $\mathbf{L}_2$ error. It is clear that the error decreases as the influence radius increases.

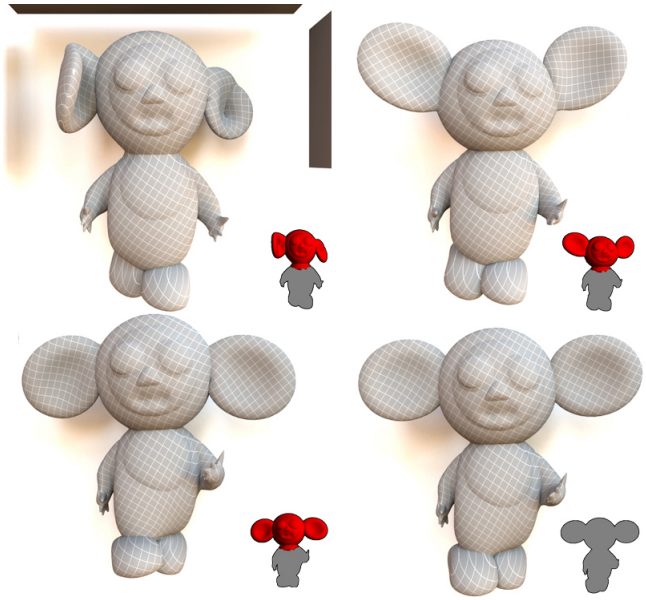


Figure 6: *In reading order, Cheb's head is simulated in full space (inset, in red) even after the walls are removed, because accelerations still produce out-of-basis deformations. Once the dynamics damp out (lower right), the subspace-only simulation re-activates.*

Cheb: The "Cheb" mesh [Baran and Popović 2007], shown in Figs. 1, 2 and 6, contains 26,652 vertices and 123,464 tetrahedra, and is rigged with 17 bones. We constructed the subspace using 38 evenly spaced samples of a walk cycle. The total rank of the subspace basis, summed over all the skeletal domains, is 301. SCC was used to resolve the predictable collisions between Cheb's feet.

As an extreme collision test, we had three walls gradually crush Cheb's head. The forces caused extreme locking in the subspace-only simulation (Fig. 2), while our simulation gracefully handled the deformation by activating full space computation in half of the vertices (Fig. 1). A large influence radius was needed in this example ($\rho = 0.39$) due to the geometric and material properties of the model: an impulse at the tip of the ear propagates quickly through the entire ear. On average, this difficult scenario was accelerated by a factor of 4.8×. Even when half of the mesh was being simulated in full space, we saw a 1.9× speedup; the other half of the mesh was simulated in the subspace for a negligible additional cost.

Dynamics were also enabled in this example, so the accelerations

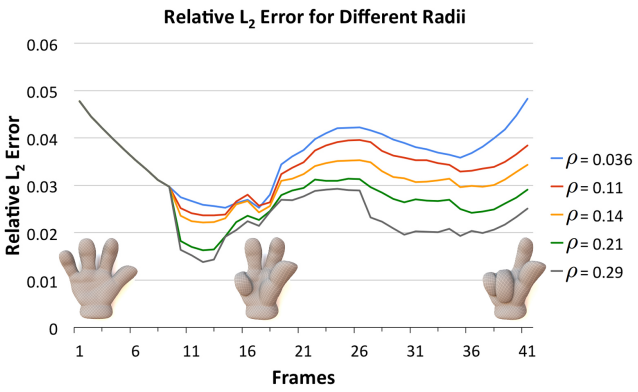
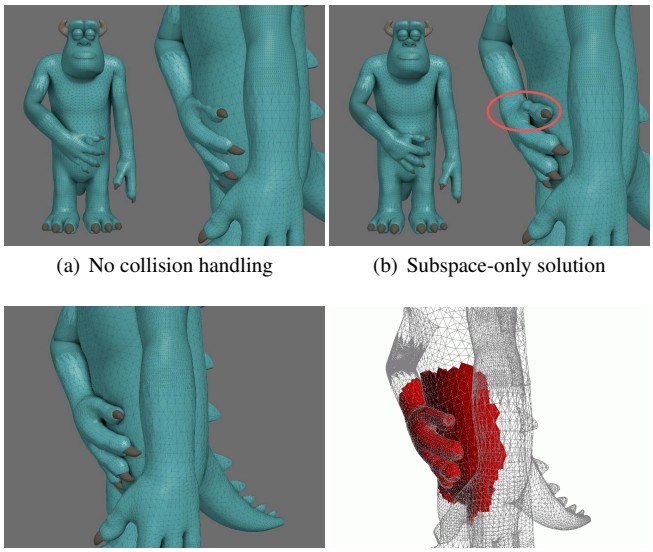


Figure 7: *The relative error of our approach using different influence radii, plotted with different colors, compared to the full simulation. Full space regions start appearing at frame 9. The relative error clearly decreases as the radius increases.*

caused by the contact forces also induced out-of-basis deformations. The dynamics oracle correctly kept the head in full space until the motion had sufficiently damped out (Fig. 6).



(a) No collision handling (b) Subspace-only solution

(c) Subspace condensation (d) Visualization of the full region

Figure 8: *Comparison of our approach with subspace-only simulation in the presence of a non-local contact. The subspace-only simulation produces a locking artifact that causes the thumb to deflate unnaturally, circled in red. Our subspace condensation technique activates a subset of full space tets (in red, lower right) that resolves the novel contact while still preserving much of the speedup from the subspace simulation.* ©*Disney/Pixar*

Sullivan: We tested our algorithm on a production model, Sullivan from *Monsters University*. The mesh contains 117,212 vertices and 611,348 tetrahedra, and is rigged with 52 bones. Similar to the hand example, the subspace is constructed from 875 snapshots obtained by actuating each bone in isolation. The total rank of the subspace is 1345. Simulations were run on an 8-core, 3.0 Ghz MacBook Pro with 16 GB of RAM using 8 threads. Local collisions such as those near the shoulders and elbows are handled using SCC.

As a normal use case, we put Sullivan through a walk cycle. The result is shown in the supplemental video. Subspace condensation

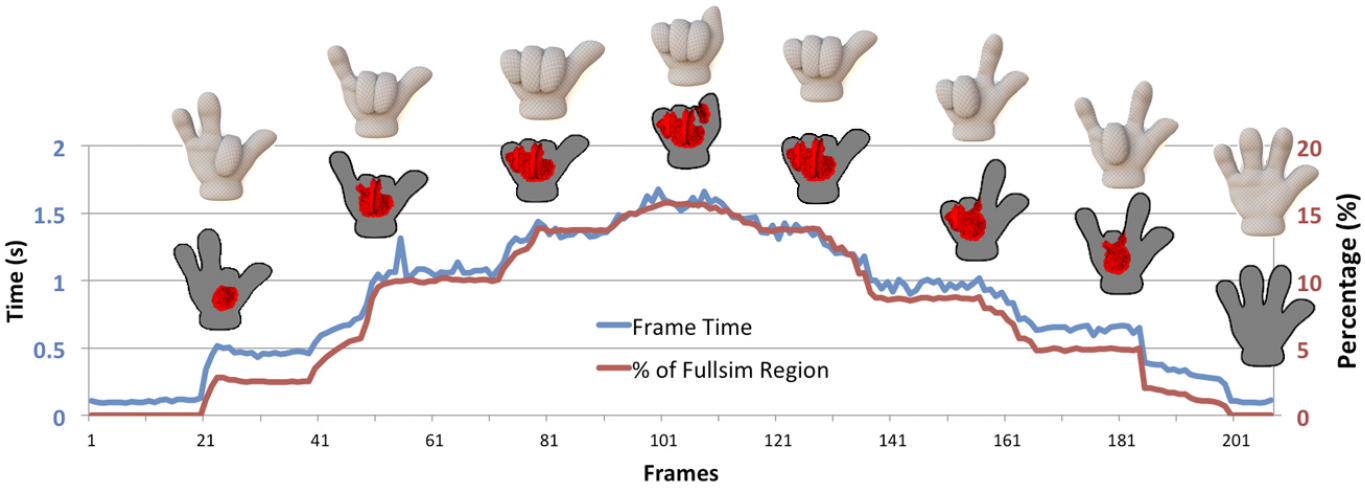


Figure 9: *Simulation time per frame for a fist clenching and unclenching. The fingers clench in a different sequence from which they unclench, so many novel collision configurations that were not seen during training are encountered. The full space regions are drawn in red in the gray silhouette images. The simulation time is clearly proportional to the size of the active full space.*

Example	Fullspace Time/Frame	Subspace Time/Frame	Best Case Speedup	Influence Radius	Average			Worst Case		
					Fullsim %	Time/Frame	Speedup	Fullsim %	Time/Frame	Speedup
Capsule	7.95 s	0.015 s	**506**×	0.25	18.6%	0.84 s	**9.4**×	50.6%	2.7 s	**3.5**×
				0.08	2.6%	0.11 s	**76**×	10.2%	0.42 s	**22**×
Hand	13.54 s	0.1 s	**135**×	0.046	1.7%	0.28 s	**48**×	14.3%	1.67 s	**8.1**×
Cheb	3.69 s	0.055 s	**67**×	0.39	26%	0.77 s	**4.8**×	49.4%	1.90 s	**1.9**×
Sullivan	19.14 s	0.37 s	**52**×	0.03 (walk)	0.4%	0.83 s	**23**×	2.6%	1.37 s	**14**×
				0.048 (pat)	5.7%	1.26 s	**15**×	11.4%	2.03 s	**9.4**×

Table 2: *Performance of our subspace condensation algorithm compared to full space simulations over the entire mesh, and subspace-only simulations. On the right, we report the % of vertices that were activated in full space (Fullsim %), the time/frame and the speedup averaged over the entire sequence as well as in the worst, most complex frame. Note that Sullivan is run on a different machine from the other examples and ρ=0.03 is used for the walk sequence while ρ= 0.048 is used for the belly-pat sequence.*

is used to handle occasional collisions between the tail, feet, and thighs. Only a small influence radius ($\rho = 0.03$) is needed, and the overall performance is close to that of a subspace-only simulation.

We then ran a more challenging test where Sullivan pats his belly. Without collision handling, his right hand penetrates his body (Fig. 8(a)). The subspace-only simulation tries to resolve the collision using global deformation modes, which result in artifacts around the hand and legs (Fig. 8(b)). A slightly larger influence radius ($\rho = 0.048$) is used so that the entire hand is simulated in full space (Figs. 8(c) and 8(d)). Our method correctly handles this novel, non-local self-collision while still maintaining performance that is on the same order as the subspace-only simulation.

6 Limitations and Future Work

We have presented an efficient algorithm that addresses a key limitation of subspace simulations. When an out-of-basis event is encountered, we activate full space computation on-the-fly in the neighborhood of the event. Our method is fast, generic and applies to non-linear materials.

While our condensation algorithm allows subspace acceleration to be applied to new scenarios, some of the usual limitations of subspace methods still remain. If there are global deformations that are not well-captured by the subspace, some artifacts can appear. For example, collisions that produced near-rigid translations of the capsule sometimes produced slight swimming in the checkerboard texture if an equivalent quasi-translation was not captured by the basis. So, while basis construction becomes less onerous with our method, the subspace must still be constructed with care.

Our method can support any oracle, but the quality of that oracle will directly determine the amount of acceleration experienced by the simulation. We have proposed two oracles based on spherical distance and global velocity and acceleration. More sophisticated approaches are almost certainly possible, and their design is a direction for future work.

We used penalty forces to resolve collisions, but full space adaptivity opens the door to constraint mechanisms that subspace methods have had trouble with in the past. The degrees of freedom needed for hard constraints can now be added on the fly, and enable such phenomena as adhesive contact [Gascón et al. 2010].

Finally, the coupling mechanism at the core of our method arises from applying a projection to a specific matrix partitioning. This projected partition approach could be applied in other coupling contexts as well, such solid to fluid coupling. Consequently, subspace condensation may provide a natural method of coupling subspace deformation and subspace fluid simulation [Wicke et al. 2009; Kim and Delaney 2013] methods.

Acknowledgements

The authors would like to thank the anonymous reviewers for their thorough and conscientious feedback. YT and TK are supported

by a National Science Foundation CAREER award (IIS-1253948). 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. Graph. 27*, 5 (Dec.), 165.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. In *ACM Trans. Graph.*, vol. 26, 72.

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

BATHE, K.-J. 2007. *Finite Element Procedures*. Prentice Hall.

BRO-NIELSEN, M., AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer graphics forum*, vol. 15, Wiley Online Library, 57–66.

GAO, M., MITCHELL, N., AND SIFAKIS, E. 2014. Steklov-poincaré skinning. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 139–148.

GASCÓN, J., ZURDO, J. S., AND OTADUY, M. A. 2010. Constraint-based simulation of adhesive contact. In *Proc. of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

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.

GUENNEBAUD, G., JACOB, B., ET AL., 2010. Eigen v3. http://eigen.tuxfamily.org.

GUYAN, R. J. 1965. Reduction of stiffness and mass matrices. *AIAA journal 3*, 2, 380–380.

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

HARMON, D., AND ZORIN, D. 2013. Subspace integration with local deformations. *ACM Trans. Graph. 32*, 4, 107.

HAUSER, K. K., SHEN, C., AND O'BRIEN, J. F. 2003. Interactive deformation using modal analysis with constraints. In *Graphics Interface*, vol. 3, 16–17.

IRONS, B. 1965. Structural eigenvalue problems-elimination of unwanted variables. *AIAA journal 3*, 5, 961–962.

JAMES, D. L., AND PAI, D. K. 2003. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph. 22*, 1 (Jan.), 47–82.

KAVAN, L., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. 2007. Skinning with dual quaternions. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, ACM, 39–46.

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

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

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.

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.

KRYSL, P., LALL, S., AND MARSDEN, J. E. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *Int. J. Numer. Meth. Eng. 51*, 479–504.

LEUNG, A. Y.-T. 1978. An accurate method of dynamic condensation in structural analysis. *Int. J. Numer. Meth. Eng. 12*, 11, 1705–1715.

LI, S., HUANG, J., DE GOES, F., JIN, X., BAO, H., AND DESBRUN, M. 2014. Space-time editing of elastic motion through material optimization and reduction. *ACM Transactions on Graphics 33*, 4.

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*.

PAZ, M. 1989. Modified dynamic condensation method. *Journal of Structural Engineering 115*, 1, 234–238.

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

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.

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

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

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. In *ACM Trans. Graph.*, vol. 28, ACM, 39.

WILSON, E. L. 1974. The static condensation algorithm. *Int. J. Numer. Meth. Eng. 8*, 1, 198–203.

XU, H., LI, Y., CHEN, Y., AND BARBIC, J. 2014. Interactive material design using model reduction. *ACM Trans. on Graphics*.

XU, W., UMENTANI, N., CHAO, Q., MAO, J., JIN, X., AND TONG, X. 2014. Sensitivity-optimized rigging for example-based real-time clothing synthesis. *ACM Trans. Graph. 33*, 4 (July), 107:1–107:11.

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*.