Efficient Simulation of Secondary Motion in Rig-Space

Fabian Hahn[1,2] Bernhard Thomaszewski[2] Stelian Coros[2] Robert W. Sumner[2] Markus Gross[1,2]

[1]ETH Zurich [2]Disney Research Zurich

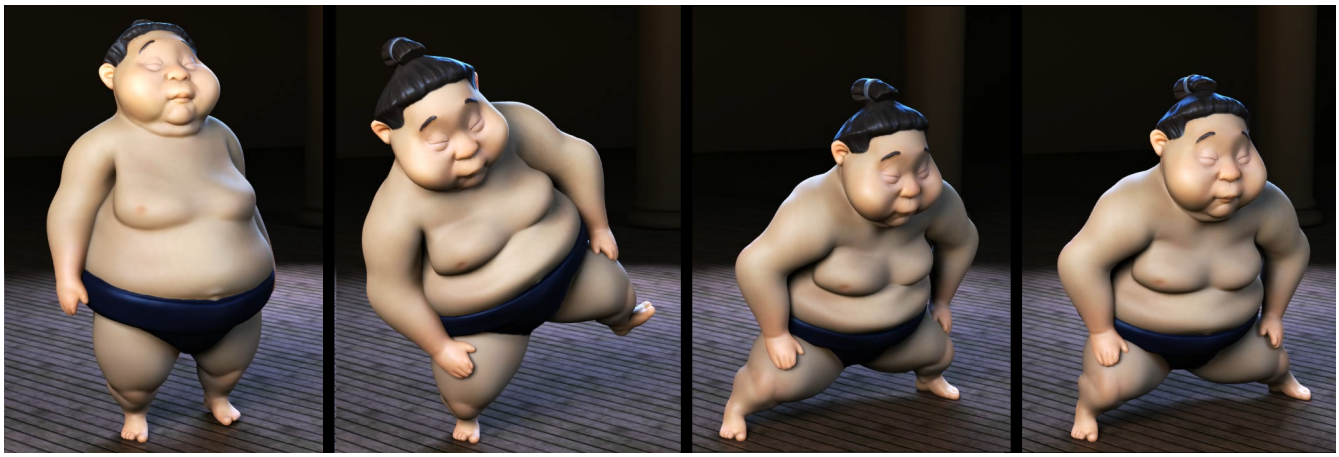

Figure 1: *Our method augments hand-crafted character animations such as this sumo wrestler with high-quality secondary motion, using an efficient rig-space simulation method.*

Abstract

We present an efficient method for augmenting keyframed character animations with physically-simulated secondary motion. Our method achieves a performance improvement of one to two orders of magnitude over previous work without compromising on quality. This performance is based on a linearized formulation of rig-space dynamics that uses only rig parameters as degrees of freedom, a physics-based volumetric skinning method that allows our method to predict the motion of internal vertices solely from deformations of the surface, as well as a deferred Jacobian update scheme that drastically reduces the number of required rig evaluations. We demonstrate the performance of our method by comparing it to previous work and showcase its potential on a production-quality character rig.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: physically-based simulation, animation control

1 Introduction

Creating believable and compelling character motions is arguably the central challenge in animated movie productions. While manually posing a character for each animation keyframe allows artists to create very expressive animations, this process is tedious when it comes to creating secondary motion such as the bulging of muscles or jiggling of fat. Hahn et al. [2012] recently presented *rig-space physics*, a method to augment keyframed animations with automatically computed secondary motion. The basic idea of rig-space physics is to use physics-based simulation in rig-space, the character's space of motion. As a key advantage over conventional physics-based simulation, the results of these simulations are animation curves that can be easily edited by artists. But while rig-space physics can automatically generate secondary motion with high visual quality, it entails a significant computational burden that slows production and prohibits its use in interactive environments.

In this paper, we present a method that offers a significant computational improvement over the work of Hahn et al. [2012] while maintaining the same level of quality. This advance is made possible by three main contributions:

- a linearized formulation of rig-space dynamics using rig parameters as the only degrees of freedom,
- a physics-based volumetric skinning method that allows our algorithm to compute the position of internal vertices solely from the surface vertices, and
- a deferred Jacobian evaluation scheme that significantly reduces the number of required rig evaluations.

Taken together, these contributions allow a performance improvement of one to two orders of magnitude over the original rig-space physics method on production-quality rigs, as shown in Fig. 1.

2 Related Work

Designing and animating digital characters is a central problem in computer graphics. We refer the interested reader to the recent survey by McLaughlin et al. [2011] for an overview of the many challenges involved in this task. Here, we focus on existing work related to the problem of creating secondary motions.

Rigging and Keyframe Animation Before characters can be animated, they first have to be modeled and rigged. During modeling, artists define the surface mesh of a character, and the rigging stage requires artists to specify how this surface mesh deforms as a function of a relatively small number of rig parameters. The map between the rig parameters and the deformation of the surface mesh can be defined using a variety of different techniques: linear blend or dual quaternion skinning [Magnenat-Thalmann et al. 1989; Kavan et al. 2008], wire deformations [Singh and Fiume 1998] or blend shapes [Lewis et al. 2000; Sloan et al. 2001], to name a few. Many of these techniques are complementary, and there is no single best solution for all applications. Furthermore, the choice of which of the techniques are eventually used also depends on other factors such as personal preference. In order to afford a maximum level of generality, we follow Hahn et al. [2012] and do not make any assumptions about the underlying rig. Consequently, we evaluate the rig and its derivatives through function calls to the modeling/animation software.

Deformable Models Physics-based simulation is a natural choice for creating secondary motion effects such as flesh and fat jiggling as a character moves. Since the pioneering work of Terzopoulos et al. [1987], many simulation methods that can potentially be used for this purpose have been introduced. A comprehensive review of these works is outside the scope of this manuscript, but the survey article of Nealen et al. [2006] provides more details on this topic.

Most of the methods for simulating volumetric objects require the simulation domain to be spatially discretized into tetrahedrons [Irving et al. 2004]. The characters used in animation environments are typically represented by surface meshes only, but there are many well-established tools such as *Tetgen*, *NetGen*, and *gmsh* for generating tetrahedral meshes from boundary representations. A more fundamental difference between physics-based simulation and character animation is that simulations endow each vertex of the mesh with individual degrees of freedom. To the extent that the motion must obey physics, the vertices are thus free to move independently from each other. This setup is in stark contrast to character animation, where the high-resolution surface mesh is constrained to deform only in the subspace defined by the rig. This representational mismatch typically implies that simulation must take place at a later stage of the animation pipeline and that results cannot easily be edited by animators.

The rig-space physics method of Hahn et al. [2012] was specifically designed to bypass this challenge, but it still needs to compute the motion of the internal vertices. This entails the solution of large systems of equations and, consequently, leads to high computational costs. Although there are simulation methods that do not require a volumetric representation, these are not without limitations. Shell models [Grinspun et al. 2003], for instance, can, in principle, be used to compute secondary motions on the character's surface. However, shell models lack volume preservation by definition and thus cannot account for the natural bulging of flesh and other soft tissue. As another alternative, the boundary element method [James and Pai 1999] condenses a volumetric problem to one with degrees of freedom only on the surface. However, this method only works well for linear problems and is therefore not attractive for modeling the highly nonlinear deformations exhibited by production-quality characters.

To alleviate these limitations, we aim to formulate an explicit, linear map that returns the position of interior vertices as a function of the configuration of the surface mesh. In effect, the deformation of the rig is automatically propagated everywhere in the interior of the simulation mesh, allowing for a very efficient implementation of subspace physics.

Subspace Physics Reduced model methods for deformable objects are typically used to improve simulation speed. The underlying idea is to formulate the equations of motion in a low-dimensional subspace onto which the full-dimensional simulation model is projected. These subspaces can be defined by applying dimensionality reduction on sequences of meshes obtained through full simulations [James and Fatahalian 2003], by embedding objects in low-resolution lattices [Faloutsos et al. 1997], by analyzing the vibration modes of an object [Barbič et al. 2009], or by using dual quaternion skinning to express the deformation of an object as a function of a small number of reference frames [Gilles et al. 2011]. While these methods are typically optimized for efficiency, rig-space physics is designed to operate in the deformation subspaces defined by arbitrary animation rigs. This generality, however, comes at a heavy computational price, which we aim to significantly lower with the method we propose.

Skinning One of the key contributions of our method is that it enables the use of a deformation energy defined on a volumetric mesh, but without the need for additional internal degrees of freedom. We construct an explicit, example-based linear map that outputs the configuration of the internal vertices as a function of the surface mesh of the character. This approach is inspired by existing methods that compute skinning weights to map the motion of a set of coordinate frames onto a surface mesh [Magnenat-Thalmann et al. 1989]. There are many methods that aim to improve the quality of skinning. Multi-linear methods [Wang and Phillips 2002] use additional weights to improve the quality, as does the recent method of Jacobson et al. [2012]. Kavan et al. [2008] show that nonlinear skinning formulations can also lead to better quality, but our goal is to construct a linear map between surface and internal vertices. Another group of methods improves skinning using example shapes [Lewis et al. 2000; Sloan et al. 2001; Kry et al. 2002], while other methods automatically compute skinning weights by optimizing for smoothness properties [Baran and Popović 2007; Jacobson and Sorkine 2011]. The recent method by Kavan et al. [2012] automatically computes optimized skinning weights by minimizing an elastic energy, which is similar in spirit to our approach. However, our skinning method does not try to alter the deformation of the surface mesh but defines the behavior of the interior.

Other research [James and Twigg 2005; Hasler et al. 2010; Kavan et al. 2010; Le and Deng 2012] specifically targets the problem of skinning animations, which entails finding transformations and corresponding skinning weights to best approximate a sequence of deforming meshes. Our work is closely related to these approaches. We first obtain a set of example deformations for the tetrahedral mesh by using physics-based simulation on a small number of artist-generated character poses. We then optimize for a sparse set of skinning weights that best explains the behavior of the internal vertices through the surface deformation.

Other methods to compute skinning weights create a low-resolution cage from the surface mesh and compute, for each internal vertex, the harmonic [Joshi et al. 2007], mean-value [Ju et al. 2005] or Green coordinates [Lipman et al. 2008]. While these methods lead to smooth deformation fields, they are not without drawbacks. Besides the fact that an artist needs to model and rig the cage, harmonic coordinates are expensive to compute, mean-value coordinates can lead to non-conformal interpolation, and the geometry interpolated using Green coordinates is not guaranteed to remain inside the control cage, which, for our problem setting, would result in inverted tetrahedrons. In addition, the deformation fields generated with these methods are disconnected from the elastic model used to represent the characters. We therefore resort to an example-based skinning method whose resulting deformation fields reflect the nature of the underlying material.

3 Foundations

The input to our method is a rigged character, which is defined by a surface mesh with n_s vertices $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_{n_s})^t$ and an abstract mapping

$$\mathbf{p} \rightarrow \mathbf{s}(\mathbf{p}) \tag{1}$$

from a set of rig parameters $\mathbf{p}$ to corresponding deformed surface positions $\mathbf{s}(\mathbf{p})$.

As a basis for simulating secondary motion in rig space, we assume that the character can be modeled as an elastic solid. We represent this solid by a volumetric mesh with n_e tetrahedral elements and n_v vertices, each of which carries a mass m_i. The deformed and undeformed positions of the vertices are stored in vectors $\mathbf{x} \in \mathbb{R}^{3n_v}$, respectively $\mathbf{X} \in \mathbb{R}^{3n_v}$. We furthermore decompose the deformed positions into sets of n_s surface vertices $\mathbf{s}_i$ whose positions are directly controlled by the rig, and n_i internal vertices $\mathbf{q}_i$.

The behavior of the solid is governed by an elastic energy $W(\mathbf{X}, \mathbf{x})$, whose precise form is defined by a deformation measure and a material law. Letting $\mathbf{x}^e = (\mathbf{x}_1^e, \ldots \mathbf{x}_4^e)$ and $\mathbf{X}^e = (\mathbf{X}_1^e, \ldots \mathbf{X}_4^e)$ denote the deformed, respectively undeformed positions of a given tetrahedral element, we compute its deformation gradient as $\mathbf{F} = \mathbf{d}\mathbf{D}^{-1}$, where $\mathbf{d}$ and $\mathbf{D}$ are (3×3) matrices whose columns hold deformed and undeformed edge vectors $\mathbf{x}_i^e - \mathbf{x}_1^e$, respectively $\mathbf{X}_i^e - \mathbf{X}_1^e$ for $2 \leq i \leq 4$. Using a modified St. Venant-Kirchhoff material as described by Martin et al. [2011], the elastic energy density per element is obtained as

$$W^e(\mathbf{x}^e) = \tfrac{1}{2}\mu||\mathbf{E}||_F + \lambda\left(1 - \frac{V^e}{V_0^e}\right) , \tag{2}$$

where $||\cdot||_F$ is the Frobenius norm, $\mathbf{E} = \frac{1}{2}(\mathbf{F}^t\mathbf{F} - \mathbf{I})$ is the Green strain, V^e and V_0^e are the deformed respectively undeformed volumes of the element, and μ, λ are material parameters. The elastic energy of the entire solid is obtained by summing up elemental contributions as $W = \sum_e W^e \cdot V_0^e$. With the undeformed configuration fixed, the elastic energy is a function of only the deformed surface and internal vertices, $W = W(\mathbf{s}, \mathbf{q})$.

Assuming that a subset $\mathbf{p}_s$ of the rig parameters is scripted over time as part of an input animation, the goal is to automatically compute the motion of the remaining rig parameters. Adopting the implicit Euler method, Hahn et al. [2012] solve time-stepping by minimizing the nonlinear functional

$$H(\mathbf{p}, \mathbf{q}) = \frac{h^2}{2}\left(\mathbf{a}_s^t\mathbf{M}_s\mathbf{a}_s + \mathbf{a}_q^t\mathbf{M}_q\mathbf{a}_q\right) + W(\mathbf{s}(\mathbf{p}), \mathbf{q}) \tag{3}$$

with respect to the free rig parameters $\mathbf{p}_f$ and internal vertex positions $\mathbf{q}$. In the above expression, $\mathbf{M}_s$ and $\mathbf{M}_q$ are diagonal mass matrices and the nodal accelerations are defined as

$$\mathbf{a}_s = \tfrac{1}{h^2}(\mathbf{s}(\mathbf{p}) - 2\mathbf{s}_n + \mathbf{s}_{n-1}) , \quad \mathbf{a}_q = \tfrac{1}{h^2}(\mathbf{q} - 2\mathbf{q}_n + \mathbf{q}_{n-1}) ,$$

where the subscripts refers to the time step index.

As demonstrated by Hahn et al. [2012], the above formulation affords high-quality simulations of secondary motion and other physics-based detail in rig-space. However, the resulting algorithm is computationally intensive for two primary reasons. First, minimizing (3) with a Newton-Raphson scheme requires first and second derivatives of the rig $\mathbf{s}$ with respect to its parameters $\mathbf{p}$. Since the rig is generally not available in analytic form, these derivatives have to be estimated using finite differences for each iteration of the solver. Second, the dimension of the resulting system is comparatively high, considering that only the free rig parameters are required. In the next section, we describe a method that greatly accelerates computations without compromising quality.

4 Efficient Rig-Space Physics

Our goal is to develop a formulation of rig-space physics that affords the same level of quality but is significantly faster. We achieve this target by establishing a linearized formulation, eliminating the internal degrees of freedom using volumetric skinning, and using a deferred Jacobian evaluation.

4.1 Linear Rig Approximation

We start by linearizing the rig at the beginning of every time step as

$$\mathbf{s}(\mathbf{p}) \approx \mathbf{s}(\mathbf{p}_n) + \mathbf{J}(\mathbf{p}_n) \cdot (\mathbf{p} - \mathbf{p}_n) , \tag{4}$$

where $\mathbf{J} = \frac{\partial \mathbf{s}}{\partial \mathbf{p}}$ is the Jacobian of the rig. This simplification is reminiscent of the semi-implicit Euler scheme described by Baraff and Witkin [1998], which relies on linearized forces. However, an important difference of our approach is that we linearize the rig but not the elastic forces and, as a result, the equations of motion remain nonlinear. Using nonlinear as opposed to linearized elastic forces leads to improved stability and requires only a few evaluations of elastic energy gradients and Hessians, which is significantly faster than evaluating $\mathbf{J}$. As a direct computational advantage of the rig linearization, we need to evaluate the Jacobian only once per time step and, moreover, second order derivatives of the rig vanish altogether.

4.2 Physics-based Volumetric Skinning

In the original rig-space physics method, the problem variables consist of the rig parameters and the position of the internal vertices. Besides the fact that the internal vertices contribute significantly more degrees of freedom than the rig parameters (≈ 100 rig parameters vs. several thousand internal vertices), the internal vertices serve only a helper role in the computation of the internal energy for a given set of rig parameters. They are not visible in the resulting animation and of no interest to the artist. We would like to establish a formulation in which only the truly relevant variables, namely the rig parameters, are exposed as degrees of freedom.

To this end, we start by assuming that the positions of the internal vertices are always defined by the boundary vertices through static equilibrium conditions:

$$\mathbf{q}(\mathbf{p}) = \arg\min_{\tilde{\mathbf{q}}} W(\mathbf{s}(\mathbf{p}), \tilde{\mathbf{q}}) . \tag{5}$$

While this formula defines a unique mapping from rig parameters to internal vertices, the corresponding function is implicit: it requires minimizing the elastic energy and thus solving a set of nonlinear equations. Since doing so would be computationally expensive, our goal is to approximate this implicit nonlinear map by an explicit linear function.

One option is to use cage-based deformation techniques such as Harmonic coordinates [Joshi et al. 2007] or Green coordinates [Lipman et al. 2008] that are used for deforming a high resolution embedded surface with an enveloping mesh. There are, however, two drawbacks with these approaches. First, an artist needs to design and rig a cage since there are no reliable automatic methods for this task. Second, while the deformation field inside the cage is smooth, the corresponding vertex positions will generally be far from their equilibrium positions as dictated by the underlying elastic material. This disparity leads to an overestimation of the elastic energy and, in turn, severely affects the dynamics of the character.

In contrast to typical cage-based modeling problems, we have explicit knowledge about how the internal deformation field should

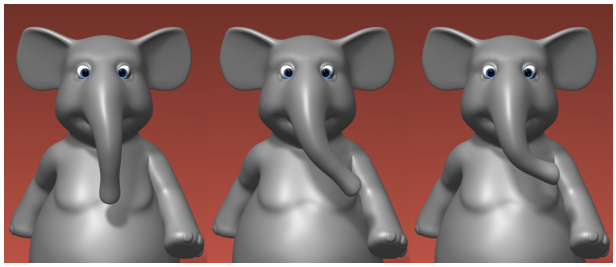

Figure 2: *An impulse vector applied along the horizontal image axis results in a swinging motion for the elephant's trunk, providing us with a sequence of surface deformations and corresponding internal deformations.*

evolve as a function of the surface mesh. Namely, for every surface configuration, we can compute the internal vertex positions by minimizing a nonlinear elastic energy. This observation motivates an example-based approach in order to compute an optimal linear approximation to the internal deformation field. We first describe how to generate a set of example poses and then explain how to compute the linear map.

Generating Example-Poses We assume that there is a small set of about five to ten artist-generated poses that are representative of the typical range of motion during animation. In a production environment, such poses are typically created as a means of testing the character during the rigging stage and are referred to as *calisthenics*. Given these basic poses, we generate an augmented example set by applying a small number of impulse vectors to the surface of the character. Each of the impulse vectors defines initial velocities for the character that we use to perform a few steps of dynamic simulation by solving (3) for the free rig parameters as well as the corresponding equilibrium positions of the internal vertices. The result of this process, which we refer to as *shaking*, is a sequence of surface positions and corresponding internal deformation which we add to the example set. In this way, we can generate a wider range of poses that also reflects the influence of the simulated rig parameters $\mathbf{p}_f$ on the internal vertices (Fig. 2).

Example-Based Skinning The shaking process provides us with a set of m deformed configurations $\mathbf{x}^e = \{\mathbf{s}^e, \mathbf{q}^e\}$, comprising both surface and internal vertices that correspond to different poses of the character. For each internal vertex $\mathbf{q}_j$, we then seek to find weights $\mathbf{w}^j$ for the surface vertices that best explain the position of the internal vertex $\mathbf{q}_j^e$ in the m different example poses,

$$\mathbf{w}^j = \arg\min_{\tilde{\mathbf{w}}^j} \left\| \begin{bmatrix} \mathbf{s}_1^1 & \cdots & \mathbf{s}_n^1 \\ \vdots & \ddots & \vdots \\ \mathbf{s}_1^m & \cdots & \mathbf{s}_n^m \\ 1 & \cdots & 1 \end{bmatrix} \cdot \tilde{\mathbf{w}}^j - \begin{bmatrix} \mathbf{q}_j^1 \\ \vdots \\ \mathbf{q}_j^m \\ 1 \end{bmatrix} \right\|^2 . \quad (6)$$

In the above system, each row stands for three equations (x,y, and z-components), while the last row asks that weights sum to one, which ensures that the internal vertices transform correctly under rigid transformations of the surface. After solving the above problem for every vertex, we obtain a linear map between surface and internal vertices, $\mathbf{q}(\mathbf{s}) = \mathbf{W}\mathbf{s}$, where

$$\mathbf{W} = [\mathbf{w}^1, \mathbf{w}^1, \mathbf{w}^1, \ldots, \mathbf{w}^{n_q}, \mathbf{w}^{n_q}, \mathbf{w}^{n_q}]^t$$

is the skinning matrix. Due to its similarity to linear blend skinning, we refer to this map as *physics-based volumetric skinning*.

The formulation of system (6) is not yet practical. First, it allows weights to be negative, which, as reported by James and Twigg [2005], can lead to overfitting: large positive and negative weights can lead to a better fit, but outside the training data, undesirable deformations occur. Second, it assumes dense correspondences since each internal vertex can potentially be influenced by every surface point. This property deteriorates run-time performance and, as shown in Sec. 5, we also found that dense correspondences give again rise to overfitting. A practical explanation for this undesirable behavior is that dense correspondences do not respect the inherent locality of the problem: the position of the internal vertices is directly influenced only by a certain neighbor set of close-by surface vertices. Despite this locality, the dense correspondence scheme can still use remote surface vertices in order to better explain the position of a given internal vertex if the training data so happens to support this prediction. However, remote correspondences do not generalize well to data outside the training set, resulting in the aforementioned overfitting.

Ideally, we would like to choose the smallest set of close-by surface vertices that yields a robust fit to the training data. Alas, automatically computing such a neighbor set is challenging. Thresholding based on Euclidean distance is difficult to implement robustly since it is unclear how to choose the cutoff value. Internal vertices in some regions, such as the elephant's belly, can be much further away from the surface than in other regions, like the arms.

Sparse Correspondences Our solution to this problem is to ask for a sparse set of correspondences that yields a fit to the training data with a guaranteed upper-bound on the error. To this end, we augment system (6) with a sparse regularizer that penalizes the L_1-norm of the weight vector as described in [Schmidt et al. 2007], thus favoring a sparse set of correspondences. In this way, we eliminate overfitting since only those vertices are used that are actually required to explain an internal vertex's behavior. At the same time, we avoid the problem of having to heuristically determine the right sets of neighbors a priori. As another advantage, the significantly reduced neighbor set also speeds up computations at run-time.

Algorithm 1 Finding a sparse correspondence set for skinning

1: **for** all internal vertices j **do**
2: $(\mathbf{w}^j, r_0) = \text{solveNNLS}(j, \mathcal{S}_0)$
3: $r = r_0, \mathcal{S} = \mathcal{S}_0$
4: **while** $r < \delta$ **or** $r/r_0 < 1.5$ **do**
5: $(\tilde{\mathcal{S}}, \tilde{\mathbf{w}}^j) = \text{reduceCorrespondenceSet}(\mathcal{S}, \mathbf{w}^j)$
6: $\mathcal{S} = \tilde{\mathcal{S}}$
7: $(\mathbf{w}^j, r) = \text{solveL1}(j, \mathcal{S}, \tilde{\mathbf{w}}^j)$
8: **end while**
9: $(\mathbf{w}^j, r_f) = \text{solveNNLS}(j, \mathcal{S})$
10: **end for**

Our method for computing sparse weights is described in Algorithm 1. Starting from a conservative (or even full) set of correspondences $\mathcal{S}_0$, we first determine the initial error r_0 of the fit by solving system (6) subject to positivity constraints for the weights using a non-negative least squares solver (line 2) as described in [James and Twigg 2005]. We then iteratively solve the L_1-regularized version of (6) using Newton's method (line 7) and remove the surface vertices with the smallest weights from the correspondence set (line 5). The iteration is stopped whenever removing an additional vertex would lead to a residual error larger than a given threshold value δ. We recompute the final weights by solving system (6) again without the L_1-regularizer (line 9).

The result of our sparse algorithm is small sets of surface vertices and corresponding weights that explain the behavior of the internal

vertices in a robust and efficient way. As we show in Sec. 5, using sparse correspondences improves both the computational efficiency at run time and eliminates overfitting.

4.3 Deferred Jacobian Evaluation

Even when keeping the Jacobian constant per time step, its evaluation still constitutes a major part of the total computation time. Yet, due to the inherent temporal coherence in animations, the Jacobian often does not change significantly from one time step to the next. Ideally, we would like to recompute the Jacobian only when necessary. While it is, to some extent, acceptable to trade accuracy for performance, we cannot compromise on stability. We therefore need a robust indicator for evaluating the error incurred by keeping the same Jacobian over multiple time steps.

In order to quantify the error of the current approximation, we compare the end-of-time-step positions $\tilde{\mathbf{s}} = \mathbf{s}(\mathbf{p}_n) + \mathbf{J}\Delta\mathbf{p}$ predicted by the linear approximation to the actual positions $\mathbf{s}(\mathbf{p}_n + \Delta\mathbf{p})$. A natural metric for this difference is the kinetic energy due to the difference in position over the time step h,

$$\Delta E_{kin} = \tfrac{1}{2h}(\tilde{\mathbf{s}} - \mathbf{s}(\mathbf{p}_n + \Delta\mathbf{p}))^t \mathbf{M}_s (\tilde{\mathbf{s}} - \mathbf{s}(\mathbf{p}_n + \Delta\mathbf{p})) \,. \quad (7)$$

Computing this indicator requires only one rig evaluation, but it provides us with valuable information about the *linearity* of the rig around the current set of parameters and in the direction of the character's motion. Taking an optimistic approach, we can always reuse the existing Jacobian to step the rig parameters forward in time. We then evaluate the indicator and, if it signals too high a degree of nonlinearity, we roll-back to the beginning of the step, compute the Jacobian $\mathbf{J}(\mathbf{p}_n)$, and simulate again. While this approach is always robust and efficient in many cases, animations with rapid motion and extreme deformations can lead to an excessive number of roll-backs, effectively undoing the advantage of deferred Jacobian evaluation. In order to decrease the number of such roll-backs, we use an additional indicator that estimates the linearity of the rig (in the relevant direction) without requiring a full simulation step: instead of solving for $\Delta\mathbf{p}$ first, we simply estimate the end-of-time-step parameters $\tilde{\mathbf{p}}$ using an extrapolation of past states as

$$\tilde{\mathbf{p}} = \mathbf{p}_n + \tfrac{1}{h}(\mathbf{p}_n - \mathbf{p}_{n-1}) \,.$$

As shown in Sec. 5, this indicator leads to significantly fewer roll-backs while limiting the number of Jacobian reevaluations.

4.4 Implementation

With our physics-based skinning method, the functional (3) now only depends on the free rig parameters: $H = H(\mathbf{p})$. We perform time-stepping by minimizing $H(\mathbf{p})$ using Newton's method, which requires the gradient and Hessian of H. For convenience, we include the relevant formulas here. We emphasize again that, while each iteration requires the reevaluation of the elastic energy gradient and Hessian, the rig Jacobian stays constant.

The gradient of H is obtained as

$$\frac{\partial}{\partial \mathbf{p}} H(\mathbf{p}) = \mathbf{J}^t \left(\mathbf{M}_s \mathbf{a}_s + \frac{\partial W}{\partial \mathbf{s}} + \mathbf{W}^t \left(\mathbf{M}_q \mathbf{a}_q + \frac{\partial W}{\partial \mathbf{q}} \right) \right) \,,$$

and the Hessian follows as

$$\frac{\partial^2 H(\mathbf{p})}{\partial \mathbf{p}^2} = \mathbf{J}^t \left(\frac{1}{h^2}\mathbf{M}_s + \frac{\partial^2 W}{\partial \mathbf{s}^2} + \mathbf{W}^t \left(\frac{1}{h^2}\mathbf{M}_q + \frac{\partial^2 W}{\partial \mathbf{q}^2} \right) \mathbf{W} \right) \mathbf{J} + \mathbf{J}^t \left(\mathbf{W}^t \frac{\partial^2 W}{\partial \mathbf{q} \partial \mathbf{s}} + \frac{\partial^2 W}{\partial \mathbf{s} \partial \mathbf{q}} \mathbf{W} \right) \mathbf{J} \,.$$

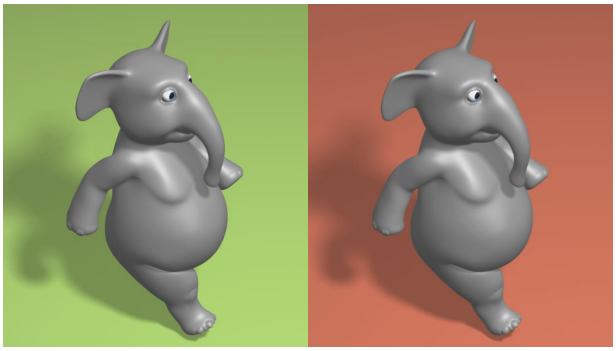

Figure 3: *A visual comparison between rig-space physics [Hahn et al. 2012] and our method for a selected frame. The average/maximum vertex error is below 0.08%/1.4% for all frames.*

5 Results and Discussion

In this section, we present results of our method and compare its performance to previous work. We also validate the physics-based skinning method by comparing it to a state-of-the-art alternative approach.

For a fair comparison with rig-space physics [Hahn et al. 2012], we apply our method to the elephant walk cycle animation from the original work, using the same parameter values as reported in the corresponding paper. To be consistent with [Hahn et al. 2012], we simulate the trunk and the belly of the elephant separately, which are controlled through 13 (trunk) and respectively 36 (belly) rig parameters. Taking the animation produced by rig-space physics as the ground truth, we compare the simulation time as well as the difference in geometric deformation to our method using several different options.

The results of these comparisons are summarized in Table 1 and a visual comparison for an exemplary frame is shown in Fig. 3. As can be seen from these data, our linearized formulation (row 2) is already significantly faster than the reference method (row 1). Yet, our physics-based volumetric skinning and its associated dimension reduction achieves another drastic speedup. We point out that, despite these significant accelerations, the quality of the animation remains the same: visually, the differences between the original method and our approach are imperceptible, as the average/maximum difference in vertex position for the belly is below 0.08%/1.4% of the character's height for all frames.

Table 1 also indicates that our deferred Jacobian evaluation scheme (rows 4 and 5) is very effective for the belly of the elephant, whereas there is no speedup for the trunk. This result occurs because the rig is mostly linear in the region of the belly and our method can thus leverage the full potential of deferred Jacobian evaluation. For the trunk, however, the rig is based on a nonlinear wire deformer and its Jacobian changes significantly in virtually every step of the animation. Deferred Jacobian evaluation is therefore not helpful for this example, but our linearity predicator is able to detect the nonlinearity of the rig and triggers Jacobian updates for 280 out of 360 frames. In contrast, when simulating the trunk with deferred Jacobian updates but without the predictor, 281 frames had to be resimulated, resulting in a severe performance penalty.

Our second example is an animation of a sumo wrestler, *Sumone*, performing a characteristic foot stomp followed by vigorous head shaking. In addition to the usual pose controls, this character also has a total of 174 rig parameter for secondary motion, all of which we simulate using our method. An exemplary frame of this ani-

Figure 4: *An exemplary frame from an input animation without secondary motion (*left*), and with secondary motion for belly, chest, hair, and cheeks simulated using our method (*right*).*

mation is shown in Fig. 4, but the results are best viewed in the accompanying video, in which it can be seen that our method achieves lively and organic-looking motion for the character's belly, chest, cheeks, and hair. On the performance side, simulating this large number of parameters with rig-space physics is out of reach, whereas our method takes less than one second per animation frame. The timings for this animation are also listed in Table 1. Similar to the first example, our physics-based skinning offers a significant speedup, as does the deferred Jacobian evaluation.

Skinning In order to analyze the efficiency of our skinning method, we compared it to the non-negative least squares (NNLS) solver described by James and Twigg [2005]. We quantify the performance of these approaches by plotting the resulting elastic energy for the individual frames of the elephant belly animation as shown in Fig. 5.

We start by using the NNLS scheme on a large set of 500 closest surface vertices for each internal vertex. We measure the distance between internal and surface vertices in a geodesic sense by marching along the volume mesh. In this way, we ensure that a surface vertex on the belly is not erroneously quantified as close to an internal vertex of an arm.

As can be seen from Fig. 5, the elastic energy obtained with NNLS weights on this large correspondence set (red curve) is significantly higher than the ground truth (blue curve), except for the regions around the three frames which were part of the training set. Moreover, the ground truth shows a regular saw-tooth pattern, but the NNLS solution exhibits two pronounced spikes where the elastic energy is approximately five times higher than the reference value. Reducing the per-vertex correspondence set to the 20 closest surface vertices even leads to slightly worse behavior (purple curve). However, starting from the same 20 correspondences per internal vertex, our physics-based skinning method is not only able to reduce the average number of correspondences to 5.56, it also leads to a much closer tracking of the reference data. This behavior can be explained by the fact that our sparsity-based weight computation scheme only keeps correspondences that are actually required, thus eliminating overfitting. As a final comparison, using NNLS with only the 5 closest surface vertices leads to unusable behavior.

In our system, the skinning weights have to be computed only once in a preprocessing step. For the elephant mesh (1328 interior and 761 surface vertices), we used 74 poses as training data for the skinning. The initial correspondence sets of 20 surface vertices per internal vertex was reduced to an average of 5.52/5.55 in 30s/29s per

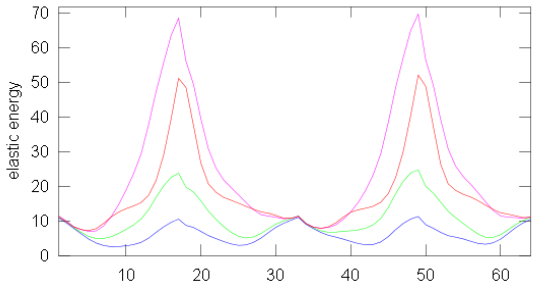

Figure 5: *Elastic energy plots for 64 frames of the elephant belly animation using static solve (blue), NNLS with 500 correspondences (red), NNLS with 20 correspondences (purple), and our physics-based skinning (green).*

Solver	Trunk		Belly		Sumone	
	t_{frame}	sp	t_{frame}	sp	t_{frame}	sp
RSP	7.24	$\times 1$	46.5	$\times 1$	—	—
IV1, JE1	0.37	$\times 19$	0.53	$\times 86$	2.82	$\times 1.0$
IV1, JE2	0.39	$\times 18$	0.39	$\times 118$	2.48	$\times 1.1$
IV2, JE1	0.12	$\times 56$	0.27	$\times 168$	1.04	$\times 2.7$
IV2, JE2	0.13	$\times 53$	0.13	$\times 348$	0.58	$\times 4.9$

Table 1: *Timings for the elephant walk cycle (*Trunk*,* Belly*) and the Sumone animation on an Intel Core i7-930 4 x 2.8Ghz using per-frame (JE1) and deferred (JE2) Jacobian evaluation, as well as static solves (IV1) and skinning (IV2) for the internal vertices.* t_{frame} *is computation time per frame in sec., and* sp *is the speedup.*

vertex for the trunk/belly, respectively. The Sumone mesh consists of 967 interior and 1302 surface vertices and we used 77 poses for skinning. The initial correspondence set of 20 surface vertices per internal vertex was reduced to 6.87 on average in 16.5s (per vertex).

6 Conclusion

We presented an efficient method to augment artist-generated keyframe animations with physically-simulated secondary motion. Our method is significantly faster than the reference solution by Hahn et al [2012], but it provides the same quality and inherits all of its benefits. The robustness and performance of our method are grounded on a linearized formulation of rig-space physics, a physics-based volumetric skinning method, as well as a deferred Jacobian evaluation scheme.

Limitations and Future Work Editing the material stiffness for individual rig parameters is not currently supported by our method. As a related challenge, it can be cumbersome to find material parameters that yield soft behavior around the rest state but do not lead to excessive deformations for fast motion. A promising direction for future work involves investigating the intuitive design and art direction of such materials. In this endeavor, our method could provide quick feedback on the outcome.

For the examples show in this paper, we used a constant time step size 0.01s, which is a fraction of the upper bound as dictated by the number of frames per second. But while parts of an animation might actually admit this maximum step size, sequences with rapid motion and large deformations will typically require smaller steps in order to maintain stability. An adaptive time stepping scheme could exploit this fact, thereby increasing robustness and efficiency.

Our system currently uses a single rest state mesh and extreme character poses can potentially lead to very distorted or even inverted elements. An interesting avenue for future work would be to investigate remeshing approaches or even meshless discretizations. Our method seems to invite such adaptive approaches since it uses only the free rig parameters as real degrees of freedom, making adaptations to the underlying mesh a lightweight process.

Acknowledgements

We would like to thank the anonymous reviewers for their useful comments and suggestions. Many thanks to Maurizio Nitti for creating the *Sumone* character.

References

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. of ACM SIGGRAPH '98*.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. In *Proc. of ACM SIGGRAPH Asia '07*.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. In *Proc. of ACM SIGGRAPH '09*.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 1997. Dynamic free-form deformations for animation synthesis. *IEEE Trans. on Visualization and Computer Graphics 3*, 3.

GILLES, B., BOUSQUET, G., FAURE, F., AND PAI, D. 2011. Frame-based elastic models. *ACM Trans. on Graphics 30*, 2.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *Proc. of Symp. on Computer Animation'03*.

HAHN, F., MARTIN, S., THOMASZEWSKI, B., SUMNER, R., COROS, S., AND GROSS, M. 2012. Rig-space physics. In *Proc. of ACM SIGGRAPH '12*.

HASLER, N., THORMÄHLEN, T., ROSENHAHN, B., AND SEIDEL, H.-P. 2010. Learning skeletons for shape and pose. In *Proc. of Symp. on Interactive 3D Graphics '10*.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. of Symp. on Computer Animation '04*.

JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. In *Proc. of ACM SIGGRAPH Asia '11*.

JACOBSON, A., BARAN, I., KAVAN, L., POPOVIĆ, J., AND SORKINE, O. 2012. Fast automatic skinning transformations.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. In *Proc. of ACM SIGGRAPH '03*.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo: accurate real time deformable objects. In *Proc. of ACM SIGGRAPH '99*.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. In *Proc. of ACM SIGGRAPH '05*.

JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. In *Proc. of ACM SIGGRAPH '07*.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. In *Proc. of ACM SIGGRAPH '05*.

KAVAN, L., AND SORKINE, O. 2012. Elasticity-inspired deformers for character articulation. In *Proc. of ACM SIGGRAPH Asia '12*.

KAVAN, L., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending.

KAVAN, L., SLOAN, P.-P., AND O'SULLIVAN, C. 2010. Fast and efficient skinning of animated meshes. In *Proc. of Eurographics '10*.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *Proc. of Symp. on Computer Animation '02*.

LE, B. H., AND DENG, Z. 2012. Smooth skinning decomposition with rigid bones. In *Proc. of ACM SIGGRAPH Asia '12*.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. of ACM SIGGRAPH '00*.

LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. In *Proc. of ACM SIGGRAPH '08*.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1989. Joint-dependent local deformations for hand animation and object grasping. In *Proc. of Graphics Interface '88*.

MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. 2011. Example-based elastic materials. In *Proc. of ACM SIGGRAPH '11*.

MCLAUGHLIN, T., CUTLER, L., AND COLEMAN, D. 2011. Character rigging, deformations, and simulations in film and game production. In *ACM SIGGRAPH 2011 Courses*.

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

SCHMIDT, M. W., FUNG, G., AND ROSALES, R. 2007. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *ECML '07*, 286—297.

SINGH, K., AND FIUME, E. L. 1998. Wires: A geometric deformation technique. In *Proc. of ACM SIGGRAPH '98*.

SLOAN, P.-P. J., ROSE, III, C. F., AND COHEN, M. F. 2001. Shape by example. In *Proc. of Symp. on Interactive 3D Graphics '01*.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proc. of ACM SIGGRAPH '87*.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proc. of Symp. on Computer Animation '02*.