Rig-Space Physics

Fabian Hahn[1,2] Sebastian Martin[2] Bernhard Thomaszewski[2] Robert Sumner[2] Stelian Coros[2]
Markus Gross[1,2]

[1]ETH Zurich [2]Disney Research Zurich

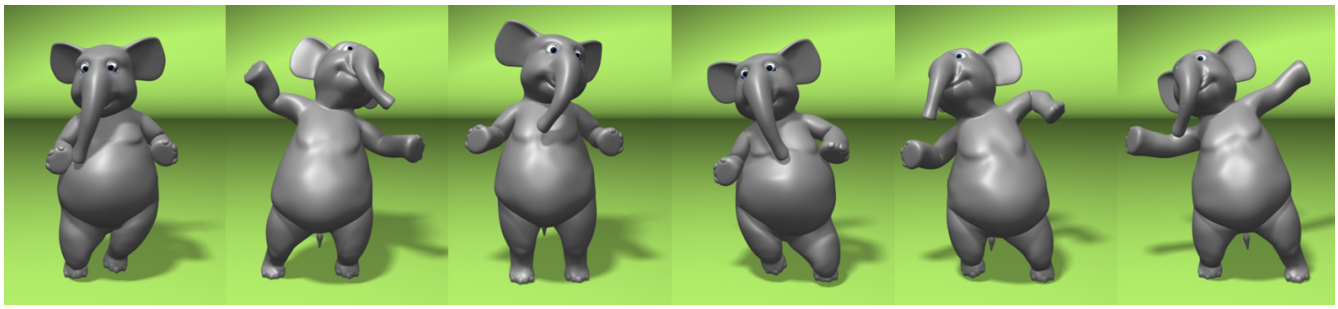

Figure 1: *A result of our method: given a character rig and a set of keyframes for some of its parameters, our method automatically produces animation curves for the remaining parameters by solving the equations of motion in the space of deformations defined by the rig. The resulting motion is physically plausible, maintains the original artistic intent, and is easily editable.*

Abstract

We present a method that brings the benefits of physics-based simulations to traditional animation pipelines. We formulate the equations of motions in the subspace of deformations defined by an animator's rig. Our framework fits seamlessly into the workflow typically employed by artists, as our output consists of animation curves that are identical in nature to the result of manual keyframing. Artists can therefore explore the full spectrum between handcrafted animation and unrestricted physical simulation. To enhance the artist's control, we provide a method that transforms stiffness values defined on rig parameters to a non-homogeneous distribution of material parameters for the underlying FEM model. In addition, we use automatically extracted high-level rig parameters to intuitively edit the results of our simulations, and also to speed up computation. To demonstrate the effectiveness of our method, we create compelling results by adding rich physical motions to coarse input animations. In the absence of artist input, we create realistic passive motion directly in rig space.

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

Links: DL PDF

http://doi.acm.org/10.1145/2185520.2185568

1 Introduction

Character animation is a vital component of contemporary computer games and film productions. For animated movies in particular, a character's movements are critical to conveying personality and style. During the rigging stage, the range of meaningful deformations for a character is carefully designed by artists in terms of a low-dimensional set of intuitive control parameters. The character's movement is determined during the animation phase, when animators set values for the control parameters over time in order to bring the character to life and make it act.

Believable and compelling animation, however, requires careful consideration of the complex physical forces involved in movement in order to give weight and substance to an otherwise empty and weightless shape [Whitaker and Halas 2002]. Manual keyframe animation affords the greatest degree of creative freedom and permits extremely expressive animation. However, grounding this expressive acting with physically realistic motion due to inertia, deformation propagation, or collision reaction can be extremely cumbersome and time consuming. Physics-based methods excel at creating such realistic effects, but are inherently difficult to control and do not respect the style of deformation chosen for the character and encoded in its rigging controls. This disconnect between the workflow used by artists (manually keyframing a small set of rig parameters) and the output of physics-based simulations (a high number of independent degrees of freedom) limits the effectiveness of physical simulation in the animation pipeline. To date, artists must choose between laboriously keyframing physical effects or employing physics-based tools that offer limited control and may not respect the envisioned range of meaningful deformations.

Our research addresses this shortcoming with a method that unifies keyframing and physical simulation to create realistic motions for rigged characters. To this end, we create an underlying representation of the character based on an elastic deformable material. We then simulate the dynamics of the character in the sub-space of deformations described by the character's rig. Our method treats all rigging controls in a unified manner and thus works with skeletons, blend shapes, spatial deformation fields, or any other rigging procedure. Moreover, the output of our system consists of animation keyframes for the rig parameters, making editing convenient for the artist.

Our approach provides artists with a tool to continuously move in the spectrum between fully controlled hand-animation and free physical simulation. We demonstrate our method as a plugin to a general-purpose 3D animation software and show results on complex characters using a variety of real-world rigging controls. Further, we extend our basic framework in the following ways:

- *Rig-space Materials*: We present a novel approach to define material stiffnesses directly on the rig parameters by inferring the required stiffness distribution on the background finite element mesh.
- *Physics-Based High-Level Rig Parameters*: Analyzing physical deformations computed with our framework yields a set of high-level rig parameters that capture the main dynamic behavior of the simulated character. This enables artists to intuitively edit the output of our system by modifying only a small number of animation curves. Creating new motions in this reduced parameter space results in significant performance gains, while maintaining the expressive power of the original set of parameters.
- *Physics-Based Inverse Kinematics*: Our framework can be easily extended to solve the inverse kinematics problem in rig space through the use of additional potential energy fields acting on the underlying deformable model.

2 Related Work

Rigging a character typically refers to the process of embedding a skeleton in a surface mesh and defining how the mesh vertices are transformed according to skeletal motions. In this paper, we use the term in a more general way to denote a nonlinear mapping between a low-dimensional space of rig parameters and a high-dimensional surface mesh. In practice, this mapping takes many forms, including classic techniques such as skeleton-based methods [Magnenat-Thalmann et al. 1989], wire deformations [Singh and Fiume 1998], nonlinear functions [Barr 1984], blend shape animation [Lewis et al. 2000; Sloan et al. 2001], and free-form deformation [Sederberg and Parry 1986], all of which are implemented in some form in modern animation software. Active research in the area of rigging focuses on problems such as automatic rig generation [Baran and Popović 2007], example-based rigging [Sumner et al. 2005; Li et al. 2010], cage-based methods [Joshi et al. 2007; Ju et al. 2008; Savoye and Franco 2010], volume preservation [Rohmer et al. 2009], and many others. Capell et al. [2002] augment purely geometric skeleton-based rigs with physics using coarse finite element simulations, while McAdams et al. [2011] obtain impressive performance using a corotational formulation on hexahedral lattices for a similar problem setting. Since such a wide variety of rigging methods exists, we design our rig-based physics method to treat the rigging system as a black box so that it can work with any of the methods mentioned above. We demonstrate results using skeletons, cage and curve deformers, and blend shape rigs.

Model reduction James and Fatahalian [2003] employ model reduction techniques on a sequence of meshes to extract deformation and illumination modes that are played back in response to user interaction. Some of the subsequent work has used this subspace to speed up collision detection for deformable models [Barbič and James 2010] or to automatically generate GUI controllers for more efficient animation of blend shape models [Seol et al. 2011]. Kim and James [2009] use an online model reduction technique to speed up the simulation of nonlinear deformable models, while Barbič and Zhao [2011] assemble larger objects from reduced simulated parts.

Faloutsos et al. [1997] propose a nonlinear freeform deformation map as a simulation subspace. The internal potential energy employed is a function of the change in rig parameter values, and does not take into account the deformations of the underlying dynamic model. It therefore does not capture coupling between the different rig parameters that could arise, for instance, from volume preservation. The proposed energy formulation is also, in general, not rotation invariant. In contrast, our formulation is grounded on continuum mechanics principles, and its well-established discretizations.

A different subspace formulation has been presented by Gilles et al. [2011], building on physical simulation in the space of dual quaternion skinning frames. This parameterization defines a rig that our method could also be applied to. However, their work focuses on efficient simulation in this specific space, while our approach trades efficiency for generality by supporting the deformation space described by arbitrary rigs.

3 Rig-Based Simulation

Input / Output Our method takes as input an arbitrary character rig, and optionally, prescribed keyframed trajectories for a subset of the rig parameters. The dynamics formulation that we propose operates on the remaining subset of "free" rig parameters, as discussed in the remainder of this chapter. After every time step, the values output for these rig parameters are recorded. This results in animation curves similar in nature to those that an artist might create through keyframing.

The input animation rig is treated as a black-box, nonlinear mapping between rig parameters $\mathbf{p}$ and the rigged surface mesh $\mathbf{s}$. In other words, we only assume to have access to the map

$$\mathbf{p} \mapsto \mathbf{s}(\mathbf{p}). \tag{1}$$

Solely requiring such a black-box map increases the generality of our approach, making it applicable to virtually any type of rig parameterization. As will be described shortly, our method requires the derivatives of the map $\frac{\partial \mathbf{s}}{\partial \mathbf{p}}$ and $\frac{\partial^2 \mathbf{s}}{\partial \mathbf{p}^2}$. If these are not provided with the rig, we estimate them using finite differences. This allows us to use our method with standard animation packages such as Autodesk Maya.

Dynamic Model In order to formulate the dynamics of deformable objects in the subspace of deformations defined by the input animation rig, we first consider the equations of motion (EOM) for deformable materials

$$\rho \ddot{\boldsymbol{x}} = \mathbf{f}(\boldsymbol{x})$$

for the continuous solution $\boldsymbol{x}(\mathbf{X}, t)$, where $\mathbf{X}$ represents the undeformed configuration, t denotes time, ρ describes the mass density of the material and $\mathbf{f}(\boldsymbol{x})$ represents the density of internal and external forces acting on the system. We choose to discretize the equations of motions in time first (see, e.g., [Krause and Walloth 2009]) and apply the implicit Euler integration scheme to obtain

$$\rho \left(\frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{h^2} - \frac{\boldsymbol{v}_n}{h} \right) = \mathbf{f}(\boldsymbol{x}_{n+1}),$$

where h is the size of the timestep. This specific choice of time integration allows us also to look at the problem in its variational form [Martin et al. 2011] as the minimization of the following nonlinear functional:

$$H[\boldsymbol{x}_{n+1}] = \frac{\rho h^2}{2} \left| \left(\frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{h^2} - \frac{\boldsymbol{v}_n}{h} \right) \right|^2 + W(\boldsymbol{x}_{n+1}), \tag{2}$$

where W represents the sum of internal and external potential energies $W_{int} + W_{ext}$ such that $-\partial_{\boldsymbol{x}} W = \mathbf{f}$. The external potential energy accounts for the effects of gravity and penalty-based collisions.

Using this variational form, we can now choose an appropriate spatial discretization to obtain the final formulation of the equations of motion for deformable objects.

Spatial Discretization To spatially discretize the variational problem (2) and make use of the provided rig parameterization (1) we first create a volumetric representation of the input character. To do so, the vertices of the rigged surface mesh $\mathbf{s}$ (or a subset, if the input mesh is too dense) serve as the boundary of a tetrahedral mesh that we generate in a preprocessing step. The resulting volumetric mesh defines a set of distinct finite elements with nodal degrees of freedom (DOFs) $\mathbf{x}$:

$$\mathbf{x} = \{\mathbf{n}\} \bigcup \{\mathbf{s}\}$$

The set $\mathbf{s}$ of nodal DOFs provides a direct mapping between the surface of the deformable object and the rig, and the internal DOFs $\mathbf{n}$ are required in order to ensure a good-quality spatial discretization irrespective of the input surface mesh that is rigged. We note that the internal DOFs are independent of the rig parameters $\mathbf{p}$, because the input rig generally deforms a surface mesh, and not an arbitrary volumetric region in space. The volumetric solution field approximating the continuous solution $\boldsymbol{x}$ is given by

$$\boldsymbol{x}(\mathbf{X}) \approx \sum_i \mathbf{x}_i N_i(\mathbf{X}) = \sum_j \mathbf{n}_j N_j(\mathbf{X}) + \sum_k \mathbf{s}_k N_k(\mathbf{X}),$$

where $N(\mathbf{X})$ are basis functions associated with the nodal DOFs.

Our formulation is independent of the FEM discretization and material model chosen, as long as the nodal degrees of freedom $\mathbf{s}$ can be incorporated. For simplicity, we use tetrahedral elements with linear basis functions in combination with standard Saint-Venant Kirchhoff or Neo-Hookean material models [Irving et al. 2004] to define the internal energy W_{int}. Specifically, this allows us to replace the continuous mass and energy densities with their discrete counterparts

$$\begin{aligned} \rho &\rightarrow \mathbf{M_n}, \mathbf{M_s} \\ W(\boldsymbol{x}(\mathbf{X})) &\rightarrow W(\mathbf{n}, \mathbf{s}), \end{aligned}$$

where standard mass lumping is used. Further, this allows us to now reformulate (2) in this discrete setting as

$$\begin{aligned} H(\mathbf{n}, \mathbf{p}) = &\frac{h^2}{2} \left(\frac{\mathbf{n} - \mathbf{n}_n}{h^2} - \frac{\mathbf{v}_n}{h} \right)^T \mathbf{M_n} \left(\frac{\mathbf{n} - \mathbf{n}_n}{h^2} - \frac{\mathbf{v}_n}{h} \right) \\ &+ \frac{h^2}{2} \left(\frac{\mathbf{s}(\mathbf{p}) - \mathbf{s}_n}{h^2} - \frac{\mathbf{w}_n}{h} \right)^T \mathbf{M_s} \left(\frac{\mathbf{s}(\mathbf{p}) - \mathbf{s}_n}{h^2} - \frac{\mathbf{w}_n}{h} \right) \\ &+ W(\mathbf{n}, \mathbf{s}(\mathbf{p})), \end{aligned} \tag{3}$$

where $\mathbf{v}$ and $\mathbf{w}$ are the velocities of $\mathbf{n}$ and $\mathbf{s}$, respectively, and the subscript n indicates the previous time step. To perform simulations in the reduced subspace provided by the rig, we minimize Equation (3) as a function of the interior vertices $\mathbf{n}$ and rig parameters $\mathbf{p}$, as described in Section 4.

4 Minimization

The problem formulation derived in the previous section allows us to describe the timestepping procedure as a minimization of the nonlinear objective function presented in Equation (3). In order to efficiently minimize this objective for each timestep, we extend a simple Newton-Raphson minimization scheme to achieve better performance for the specific setting presented here.

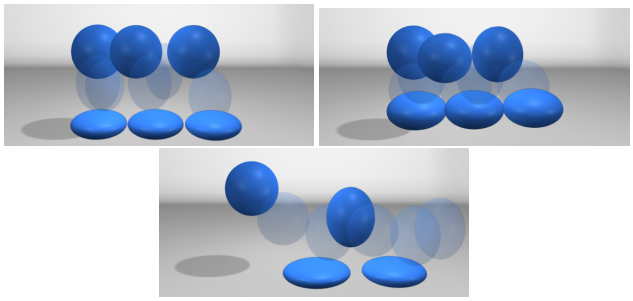

Figure 2: *A sphere with scaling and global translation rig parameters. Top left: artist-created animation, top right: constrained global translation with scaling parameters being simulated, bottom: all rig parameters being simulated.*

Derivatives Let us first state the necessary condition for a minimum as $[\partial_{\mathbf{n}} H, \partial_{\mathbf{p}} H]^T = 0$, corresponding to the conventional equations of motion for interior vertices

$$\partial_{\mathbf{n}} H = \mathbf{M_n} \left(\frac{\mathbf{n} - \mathbf{n}_n}{h^2} - \frac{\mathbf{v}_n}{h} \right) + \partial_{\mathbf{n}} W(\mathbf{n}, \mathbf{s}(\mathbf{p})) = \mathbf{0}$$

as well as rig parameters

$$\begin{aligned} \partial_{\mathbf{p}} H = &\mathbf{J_s}(\mathbf{p})^T \mathbf{M_s} \left(\frac{\mathbf{s}(\mathbf{p}) - \mathbf{s}_n}{h^2} - \frac{\mathbf{w}_n}{h} \right) \\ &+ \mathbf{J_s}(\mathbf{p})^T \partial_{\mathbf{s}} W(\mathbf{n}, \mathbf{s}(\mathbf{p})) \qquad = \mathbf{0}, \end{aligned}$$

where we make use of the notation $\mathbf{J_s}(\mathbf{p}) = \frac{\partial \mathbf{s}}{\partial \mathbf{p}}$. In order to solve this coupled system of nonlinear equations using a Newton scheme, we need to solve for correction directions $[\Delta \mathbf{n}_k, \Delta \mathbf{p}_k]^T$ in each iteration by solving the linear system

$$\begin{bmatrix} \mathbf{H_{nn}} & \mathbf{H_{np}} \\ \mathbf{H_{pn}} & \mathbf{H_{pp}} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{n}_k \\ \Delta \mathbf{p}_k \end{bmatrix} = - \begin{bmatrix} \partial_{\mathbf{n}} H \\ \partial_{\mathbf{p}} H \end{bmatrix}, \tag{4}$$

that involve second derivatives of the objective function. They are given analytically as

$$\begin{aligned} \mathbf{H_{nn}} &= \frac{1}{h^2} \mathbf{M_n} + \partial_{\mathbf{nn}} W(\mathbf{n}, \mathbf{s}(\mathbf{p})) \\ \mathbf{H_{pn}} &= \mathbf{J_s}(\mathbf{p})^T \partial_{\mathbf{sn}} W(\mathbf{n}, \mathbf{s}(\mathbf{p})) \\ \mathbf{H_{pp}} &= \mathbf{J_s}(\mathbf{p})^T \left(\frac{1}{h^2} \mathbf{M_s} + \partial_{\mathbf{ss}} W(\mathbf{n}, \mathbf{s}(\mathbf{p})) \right) \mathbf{J_s}(\mathbf{p}) \\ &\quad + \partial_{\mathbf{p}} \mathbf{J_s}(\mathbf{p})^T \mathbf{M_s} \left(\frac{\mathbf{s}(\mathbf{p} - \mathbf{s}_n}{h^2} - \frac{\mathbf{w}_n}{h} \right) \\ &\quad + \partial_{\mathbf{p}} \mathbf{J_s}(\mathbf{p})^T \partial_{\mathbf{s}} W(\mathbf{n}, \mathbf{s}(\mathbf{p})). \end{aligned} \tag{5}$$

Once the correction directions are computed, we use a line search method to find an appropriate scalar value α. The updated values for the unknowns $\mathbf{n}$ and $\mathbf{p}$ are then given by:

$$\begin{aligned} \mathbf{n}_{k+1} &= \mathbf{n}_k + \alpha_k \Delta \mathbf{n}_k \\ \mathbf{p}_{k+1} &= \mathbf{p}_k + \alpha_k \Delta \mathbf{p}_k. \end{aligned}$$

The line search parameter α_k is computed by performing cubic interpolation to previous function and gradient evaluations while satisfying the Wolfe conditions for sufficient descent, as described in [Nocedal and Wright 2006]. An example of this approach applied to a rigged sphere is shown in Figure 2.

BFGS Evaluating second derivatives analytically can be computationally expensive when a black box rig is used, as the Jacobian derivatives $\partial_{\mathbf{p}}\mathbf{J}_{\mathbf{s}}(\mathbf{p})$ need to be approximated with finite differences. This requires $\mathcal{O}(p^2)$ rig evaluations $\mathbf{s}(\mathbf{p})$ for each Newton iteration. This is currently the bottleneck in our implementation when using black box rigs evaluated by Maya.

In order to circumvent this problem, it is preferable to choose solutions that require as little use of the black box rig as possible. If analytic rig jacobians are not provided, we can use superlinearly convergent quasi-Newton methods such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [Nocedal and Wright 2006]. This method updates an approximate Hessian $\tilde{\mathbf{B}}_{k+1}$ at each Newton step by using a series of rank-2 updates involving the gradient:

$$\tilde{\mathbf{B}}_{k+1} = \tilde{\mathbf{B}}_k + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{d}_k} - \frac{\tilde{\mathbf{B}}_k\mathbf{d}_k\mathbf{d}_k^T\tilde{\mathbf{B}}_k}{\mathbf{d}_k^T\mathbf{H}_k\mathbf{d}_k},$$

where $\mathbf{d}_k = [\Delta\mathbf{n}^T, \Delta\mathbf{p}^T]^T$ and $\mathbf{y}_k$ is the gradient difference between the current and the last iteration's gradient of H. In practice, we only use BFGS to update the $H_{\mathbf{pp}}$ block of the full system Hessian. $H_{\mathbf{nn}}$ is much larger, sparse, and easy to evaluate analytically. In our experiments, the performance gained by reducing the number of calls to the black box rig outweighs the loss in convergence introduced by using approximate Hessians.

Schur Complement Solver The blocks of the system matrix in Equation (4) have different sparsity structures. While $\mathbf{H_{nn}}$ is very sparse, the blocks $\mathbf{H_{pn}}$ and $\mathbf{H_{pp}}$ are dense due to the subspace projection with $\mathbf{J_s}$. Naïvely solving this system leads to poor performance of either iterative or direct solvers.

In the spirit of [Levin et al. 2011], we first perform a Schur complement decomposition of the linear system in order to separate the dense and sparse blocks. We do this by performing a Block-Gauss-elimination of Equation (4), and thus solving the following equations in sequence

$$\begin{aligned}(\mathbf{H_{pp}} - \mathbf{H_{pn}}\mathbf{H_{nn}}^{-1}\mathbf{H_{np}})\Delta\mathbf{p} &= \partial_{\mathbf{p}}H - \mathbf{H_{pn}}\mathbf{H_{nn}}^{-1}\partial_{\mathbf{n}}H \\ \mathbf{H_{nn}}\Delta\mathbf{n} &= \partial_{\mathbf{n}}H - \mathbf{H_{np}}\Delta\mathbf{p}.\end{aligned}$$

Since $\mathbf{H_{nn}}$ can be prefactorized by a direct linear solver, the separate solves required on the relatively small number of columns of $\mathbf{H_{np}}$, $\partial_{\mathbf{n}}H$ as well as in the second equation, can be performed efficiently using back substitution. Given that the number of parameters of typical animation rigs is relatively small, the cost of the dense solve required to compute $\Delta\mathbf{p}$ is negligible.

5 Rig-Space Material Control

As discussed thus far, the input to our system consists of an animation rig and a surface mesh. The surface mesh is used to define the deformable object that drives the dynamic behavior of the simulation. The material properties of this deformable object, of course, have a large influence on the motions resulting from our system. It is therefore very important to provide an intuitive way for the user to manipulate these parameters. A simple approach would be to allow users to manually adapt the material properties of the simulated finite element mesh, which could be done, for instance, by using a painting interface. We believe that such an approach, however, does not fit well in an artist's typical workflow, as this would require an intimate knowledge of the underlying dynamical model, the meaning of the various material parameters, and the coupling between the rig and the simulated object.

To bypass this problem, we present a novel approach that allows novice users to intuitively influence the material parameters directly

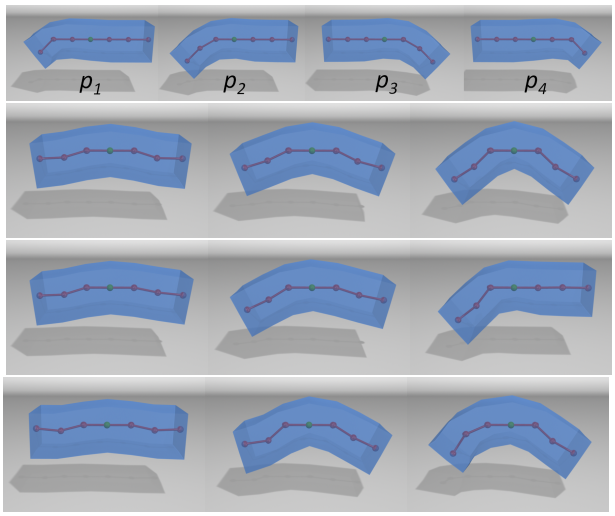

Figure 3: *Top row: An elastic bar rigged by four rig deformation modes, affecting the outmosts (p_1, p_4) and the combination of the two outer joints (p_2, p_3), respectively. Homogeneous materials result in symmetric deformations (second row), stiffening p_3 and p_4 results in overall stiffer right part of the bar (third row), while when stiffening only p_4, our approach leads to the expected behavior, despite the influence region of p_3 (last row).*

in rig-space. We allow artists to define a stiffness scale value S_i for each rig parameter $\mathbf{p}_i$, that roughly corresponds to its desired stiffness relative to the default stiffness of the homogeneous material chosen for the FEM model.

In an initial analysis step, we compute an inhomogeneous stiffness scale distribution μ_e over all simulation elements that results in the desired behavior for the rigged object. The distribution found in this step does not change our simulation framework. It only affects the stiffness of the material used during simulations.

Technically, we achieve this by treating the interior vertices $\mathbf{n}$ not as independent DOFs, but as being always in static equilibrium given the boundary conditions defined by the rigged surface points $\mathbf{s}(\mathbf{p})$. In other words, we treat every FEM node as if it was directly controlled by the rig $\mathbf{q}(\mathbf{p}) = (\mathbf{n}(\mathbf{p}), \mathbf{s}(\mathbf{p}))$. While this explicit map is only known through a nonlinear static solve, it allows us to analyze the influence of each rig parameters on the deformation of the entire FEM mesh geometry, and not only the rig-controlled boundary. We also define the derivative of this map as $\mathbf{J_q} = \frac{\partial\mathbf{q}}{\partial\mathbf{p}}$ which we evaluate with finite differences.

At the rest configuration, $\mathbf{q}(\mathbf{0})$, internal elastic forces are zero and their rate of change is described locally by the tangent stiffness matrix $\partial_{\mathbf{qq}}W$. According to Equation (5), and only considering contributions due to internal potential energy, the Hessian $\mathbf{H_{pp}}$ becomes:

$$\mathbf{H_{pp}} = \mathbf{J_q}^T\partial_{\mathbf{qq}}W_{int}\mathbf{J_q},$$

where $\partial_{\mathbf{qq}}W$ describes the energy Hessian with respect to all mesh DOFs. The columns of this Hessian describe how a change $\Delta\mathbf{p}$ introduces restoring generalized forces $\mathbf{f_p}$ that act on the rig parameters as $\mathbf{f_p} = -\mathbf{H_{pp}}\Delta\mathbf{p}$. We seek to automatically scale these restoring generalized forces using the user-specified rig-space material parameters S_i.

An alternate way of assembling the Hessian $\mathbf{H_{pp}}$ is to consider the per element energy contributions $\partial_{\mathbf{qq}}W_e$:

$$\mathbf{H_{pp}} = \mathbf{J_q}^T\left(\sum_e \partial_{\mathbf{qq}}W_e\right)\mathbf{J_q} = \sum_e\left(\mathbf{J_q}^T\partial_{\mathbf{qq}}W_e\mathbf{J_q}\right)$$

In this formulation we can now introduce the scaling factors μ_e which we aim to compute per element. These scaling factors have the effect of stiffening or softening each element individually. Formally, we ask that:

$$\mathbf{SH_{pp}} = \sum_e \mathbf{J_q}^T \partial_{\mathbf{qq}} W_e \mathbf{J_q} \cdot \mu_e, \tag{6}$$

where $\mathbf{S}$ is the diagonal scaling matrix containing the desired rig stiffness scales S_i. This is a linear system with p^2 equations and $\#e$ unknowns. We solve this system in a least squares sense using a quadratic programming approach [Nocedal and Wright 2006], where we additionally introduce inequality constraints $\mu_e \geq 0$ to obtain positive element stiffnesses and a simple L^2-regularizer proportional to $\sum_e (\mu_e - 1)^2$ that encourages the use of the default material stiffnesses.

Geometrically speaking, this approach anisotropically adapts the elastic energy landscape around the rest shape in order to reflect the desired scaling of the generalized forces. While we perform this analysis only in the neighborhood of the undeformed configuration, we have also observed intuitive resulting behaviors for large deformations. The example shown in Figure 3 demonstrates the effectiveness of this method.

6 Extensions

In the previous sections we described our simulation framework that operates directly in the space of deformations defined by an animation rig. In this section we discuss several extensions that increase the usefulness of our method.

Physics-Based High-Level Rig Parameters Many animation packages allow artists to define high-level rig parameters that combine several low-level deformers. These high-level parameters are often more intuitive to use, as they capture synergies that naturally occur in the motions being created. Our rig-space simulation framework allows us to easily create such high-level rig parameters that capture the main physical deformation modes that the underlying physical object induces on the rig parameters. The high-level parameters that we expose to the artist can be used for manual editing of physically plausible motions or as a further reduced subspace that significantly increases the speed of our simulations.

Our approach is similar in spirit to kernel PCA methods [Dambreville et al. 2006], where linear data analysis is not performed in the original vertex space, but rather in a warped or "unfolded" space. In this setting, a linear model can better capture the structure of the data. The results of our simulation in rig-space, $\mathbf{p}_i$, provide a space that already captures the nonlinearities in vertex space (i.e. $\mathbf{s}_i$). Similarly to [Krysl et al. 2001] we perform our analysis on data stemming from our rig-space simulation. Alternatively, we could use existing animations to build a model of the rig parameter coupling that artists are accustomed to using.

Since we want the undeformed configuration $\mathbf{p} = \mathbf{0}$ to always be representable in the reduced rig subspace, we choose to represent our rig parameters $\mathbf{p}$ by the span of column vectors of $\mathbf{U}$, i.e., we have a representation

$$\mathbf{p} = \mathbf{U}\mathbf{r},$$

and do not choose an affine space positioned at a mean vector as commonly done in PCA. The columns of $\mathbf{U}$ are computed straightforwardly as the right singular vectors of the data matrix $\mathbf{P} = [\mathbf{p}_1, ..., \mathbf{p}_m]$. As an illustration, different deformation modes obtained by performing this analysis on the belly motion for the elephant walk cycle are given in Figure 4. For this example, 30 parameters were initially simulated.

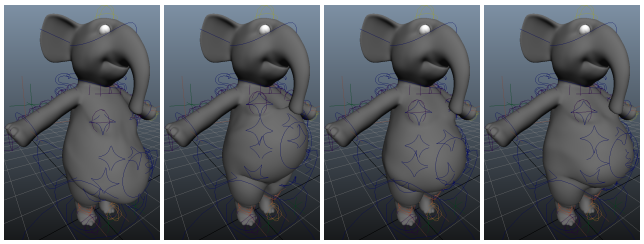

Figure 4: *PCA modes for the belly rig parameters on the walk cycle example. Left: first mode characterizing vertical motion, right: second mode for horizontal motion.*

As already mentioned, these physics-based high-level rig parameters $\mathbf{r}$ can be used as a further reduced subspace in order to speed up simulations. Incorporating this subspace into our formulation is very simple: we simply replace $\mathbf{p}$ by $\mathbf{Ur}$ in the discrete simulation energy (3) and apply the chain rule correspondingly for its derivatives.

Physics-Based Inverse Kinematics Inverse kinematics – the process of computing state variables to satisfy higher-level kinematic goals – is a well studied problem in different areas of animation [Yamane 2004; Sumner et al. 2005; Fröhlich and Botsch 2011]. A key ingredient of such systems is the definition of a quality measure that eliminates the redundancy that exists when there are more degrees of freedom than there are goals. Our physical energy model $W(\mathbf{n}, \mathbf{s}(\mathbf{p}))$ formulated in rig-space can be naturally used as such a quality measure, independently of the type of rig that is provided as input. Because the physical energy model measures internal deformation on the simulation FEM mesh, virtually all types of rigging methods can be collectively treated in a unified manner. By minimizing

$$\min_{\mathbf{p},\mathbf{n}} H(\mathbf{n}, \mathbf{s}(\mathbf{p})) + \alpha \sum_{k \in \mathrm{H}} |\mathbf{s}_k(\mathbf{p}) - \mathbf{h}_k|^2, \tag{7}$$

where H is a set of handles, we find the best configuration of rig parameters such that the surface points $\mathbf{s}_k$ are as close as possible to the goal positions $\mathbf{h}_k$. Using our flexible formulation of H, we can apply this inverse kinematics solver for static modeling but also to add dynamic effects when the momentum terms are considered.

Interior Secondary Motion The energy-based formulation defined in (3) allows us to easily switch between dynamic and static solves. The difference between the two consists of the additional momentum terms which can be omitted for static problems.

Since we decouple the treatment of the rigged vertices $\mathbf{s}(\mathbf{p})$, and the interior vertices $\mathbf{n}$, this formulation allows us to use different solvers for the two classes of degrees of freedom. It is possible, for instance, to use a static solve for the interior vertices, while using a dynamic solve for the rig parameters. In this case, during each time step, the interior points aim to reach a static equilibrium configuration due to absence of any mass and momentum. This strategy gives intuitive results for rigged objects that are meant to be more rigid. If, however, we keep the mass term for the interior points, we can achieve more complex (and interesting) secondary motion effects such as waves that propagate through the solid and affect the free DOFs of the rigged surface (cf. accompanying video). This approach works best for rigged objects that are meant to be softer. The option of using either strategy is exposed as a high-level control parameter to the users of our system.

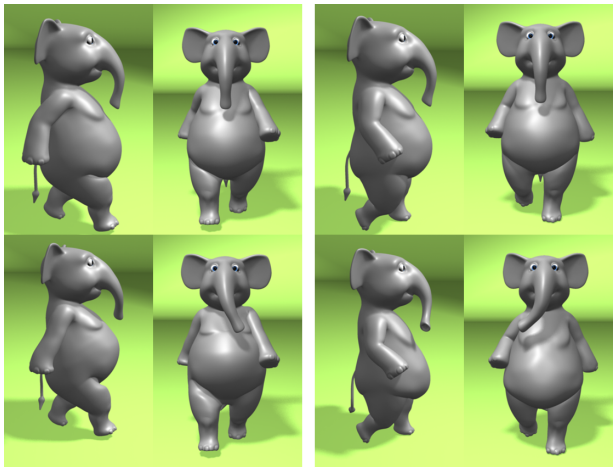

Figure 5: *The top row shows different views of two artist-rigged examples, whereas the second row shows our approach with simulated rig parameters for the belly, shoulder, trunk and tail.*

7 Results

The framework we propose provides animators with a helpful tool that can be used to enhance their existing rigged animations with physical effects and to create controllable animation of deformable objects. The approach is particularly attractive for animators since it can be seamlessly integrated in their workflow. As such, editing and post-processing of simulation results becomes as effortless as editing any other existing keyframed animation.

We demonstrate the effectiveness of our method by animating several simple rigs that are used to showcase various concepts. In addition, we animate two artist-created rigs: Prof. Peanuts and Flower. Please note that our results are best seen in the accompanying video.

To demonstrate the benefit of our approach on a practical example, we apply it to an artist-created walk cycle animation of Prof. Peanuts. Figure 5 compares our simulated animation to the coarse input animation provided by the artist. Simply specifying a subset of rig parameters to be simulated allows our framework to enrich the provided coarse, manually keyframed motion with many interesting physical details. This can be seen in the motion of the belly, trunk, ears and tail of the elephant.

Our rig-based approach also allows us to quickly iterate over animation sequences and simulate individual parts of the rig separately. Although this can potentially lead to a loss of interesting coupling between different parts of the rig, the gain in simulation speed can be significant. While using this mode of operation, the result of a previous simulation is taken as an input keyframed sequence and is treated as boundary condition for the free rig parameters. This technique has been applied for the sequence shown in Figures 1 and 5.

Rig-Space Material Control Our rig-based material model enhances the basic simulation framework presented in Section 3 by allowing the artist to specify individual stiffnesses for each rig parameter. This enriches the space of dynamic behaviors that can be generated and also allows for more accurate control over the elastic characteristics of individual parts of the rigged characters. Figure 6 shows the approach applied to the Flower rig, where neither uniformly stiff nor uniformly soft materials give satisfactory results. However, selectively weakening the blossom and the leafs, while maintaining the stiffness of the stem, leads to a desirable behavior.

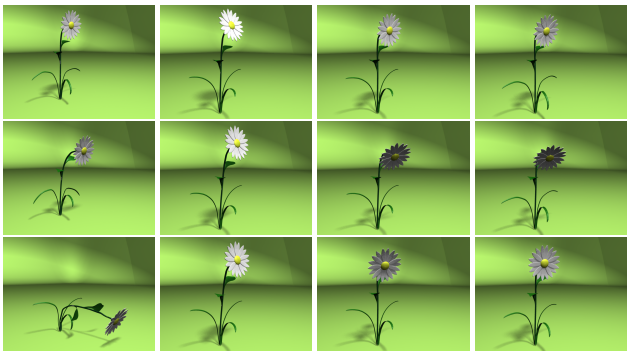

Figure 6: *In each column, the rigged flower is simulated with different rig stiffness parameters. First: homogeneous soft material, second: homogeneous stiff material, third: soft material for blossom rig, last: soft material for blossom and leaves rigs*

Physics-Based Rig Parameters To further enhance the artist's editing capabilities, and to enforce the concept of keeping the animator in the loop, our PCA-based approach automatically determines additional, physically meaningful control parameters. For the walk cycle example shown in Figure 5, the main two modes capture most of the elastic deformation of the belly during the walk. By manipulating these high-level parameters, animators can easily edit the animation curves generated by our system, in order to emphasize or reduce their effect, as illustrated in Figure 9. In our experience, this manual editing process, if not taken to an extreme, still results in plausible-looking motions.

The high-level rig parameters that we expose also provide a convenient space for our simulations to work with. As will be discussed shortly, this significantly decreases the computational overhead of our method. However, it is important to ensure that this does not negatively impact the quality of our results. To test this, we simulated the belly of the dancing elephant using the subspace built from the walk sequence. We repeated the experiment by then animating the walk sequence using the subspace obtained by simulating the full set of belly parameters for the dance motion. In both cases, the high-level rig parameters proved to generalize very well, and the visual differences between performing the simulations in the reduced space, or with the full set of parameters is almost imperceptible. This is illustrated in Figure 7.

Inverse Kinematics Defining the elastic energy on the background FEM mesh and working on abstract rig mappings $\mathbf{s}(\mathbf{p})$ allows us to perform inverse kinematics operations on arbitrary rigged objects. Figure 8 shows Prof. Peanuts performing a work out routine. This sequence was created by providing several handle trajectories, as described in Section 6. For the editing process, secondary motion can either be enabled or disabled (non-zero or zero

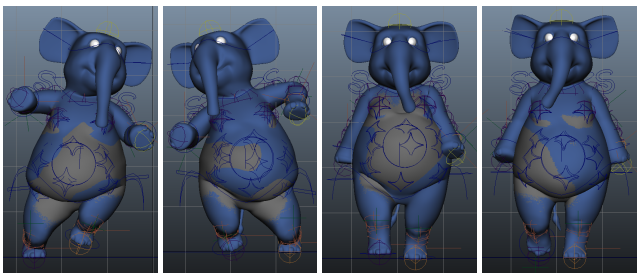

Figure 7: *The overlay of the original (blue, 30 DOFs) and reduced (grey, 6 DOFs) simulation shows the good approximation quality of our high-level controllers.*

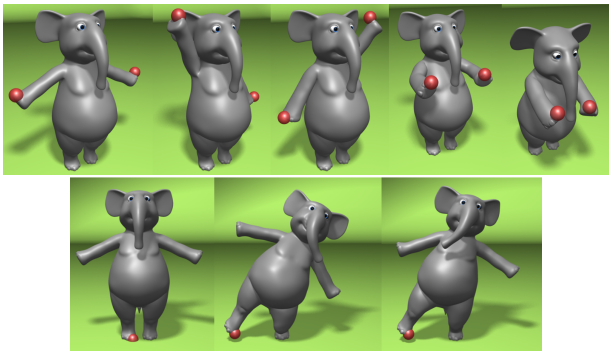

Figure 8: *Different inverse kinematics edits at deformation handles depicted as red spheres.*

mass matrices $\mathbf{M_n}$ and $\mathbf{M_s}$ in Equation (3), respectively), depending on the application.

Timing Information The performance of our method is influenced by several factors. First, the number of nodal degrees of freedom of the underlying deformable model, the number of rig parameters and the complexity of their evaluation directly influence the time required to compute the energy and its derivatives. Second, different sets of rig parameters result in energy landscapes of varying complexity. This is reflected by the varying number of iterations required for convergence.

Table 1 summarizes timing and model complexity information for the examples we ran using the BFGS solver. For the Flower and the Prof. Peanuts test sets, the Newton solver generally converged in less than 5 iterations, while the BFGS solver needed between 10 and 20 iterations to reach a predefined threshold for the gradient norm.

The computational bottleneck of our framework is caused by the black box communication interface to Maya. To assess this, we created a rig based on linear blend shapes in Maya, then replicated it in our framework with both finite difference derivatives and analytic derivatives. We used the Newton solver and enforced exactly 5 iterations per time step in all cases in order to obtain a direct performance comparison. Even with our non-optimized rig implementation that behaves identically to the one used in Maya, the processing time is reduced by a factor of 18 when using finite difference derivatives. When using analytic derivatives, the performance is further increased by a factor of more than 3. We note that this difference is likely to be more extreme as the complexity of the rigging elements is increased.

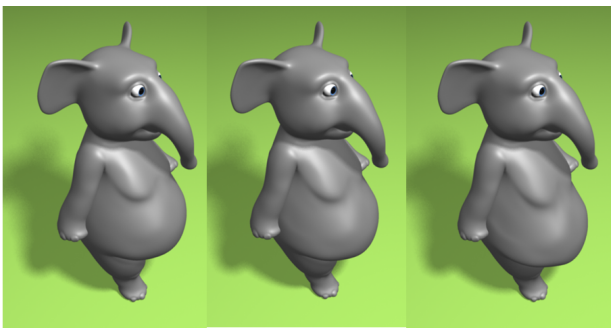

Figure 9: *A high-level rig parameter is scaled up (right) and down (left) by a factor of* 1.5 *to amplify or attenuate the belly motion.*

It is also interesting to note the large performance gain obtained when running simulations using the high-level parameters obtained through the PCA analysis. This is due to the fact that estimating the required derivatives needs $\mathcal{O}(p^2)$ rig evaluations.

Model	dim $\mathbf{p}$	dim $\mathbf{n}$	dim $\mathbf{s}$	t_{frame} (s)
elephant belly	36	3990	2253	46.47
elephant belly (PCA)	6	3990	2253	3.77
elephant trunk	13	3990	2253	7.24
elephant tail	12	3990	2253	3.45
elephant shoulder	3	3990	2253	2.51
elephant ears	2	3990	2253	1.14
flower	24	1089	987	10.54
sphere scaling	3	5430	2580	1.41
sphere passive	5	5430	2580	0.53
bar material	4	489	174	0.29
interior dynamics	12	489	174	1.22

Table 1: *Timings taken on a Intel Core i7-930 4 x 2.8Ghz*

8 Limitations and Future Work

The framework we present allows animators to exploit the benefits of dynamic simulations without any change to the typical workflow they follow when creating animations. Our method treats any animation rig as a black box, nonlinear map between a high-level set of rig parameters and the vertices of the mesh being animated. The output of our method is a set of animation curves that are virtually identical to the ones created by animators through key-framing. We show the effectiveness of our method by animating several Maya rigs. In particular, two of the examples we show consist of complex rigs that allow for a large space of deformations. To increase the usefulness of our framework, we introduce a method that allows animators to non-homogeneously change the stiffness of each rig parameter.

The animation curves produced by our system can readily be edited by artists. Scaling the animation curves in their entirety emphasizes or reduces the physical effects we introduce. We believe that investigating additional ways in which our method can enhance the productivity of artists presents a very interesting direction for future work.

Our implementation does not currently create animations at interactive rates when a large number of rig parameters is considered. To a large extent, this is caused by the relatively slow data transfer between Maya and our software. We are particularly encouraged by the effectiveness of the high-level rig parameters that we obtain by analyzing the results of our simulations. Working within this further reduced parameter space could be a key to the development of methods that allow artists to use our system at interactive or real-time rates.

Fast input motions for the rigged characters can cause numerical stability problems. This is because the boundary conditions can change drastically from one time step to the next. For our current implementation, we occasionally had to slow down the input animation, run our simulation and then re-sample the resulting motions.

The analysis performed in order to automatically control material stiffnesses only locally considers the change in internal energy caused by the various rig parameters. However, because the animation rigs we work with are nonlinear in nature, this may not lead to intuitive simulation behaviors in all regions of the rig parameter space. This limitation could be addressed by performing a global analysis, or by re-computing the material stiffnesses when the initial local model is no longer appropriate.

9 Acknowledgements

We would like to thank Derek Nowrouzezahrai, Ilya Baran, Ladislav Kavan, Joe Schmid and the anonymous reviewers for their useful comments and suggestions.
Many thanks to Maurizio Nitti for creating the rigs for *Prof. Peanuts* and *Flower*.

References

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

BARBIČ, J., AND JAMES, D. L. 2010. Subspace self-collision culling. In *Proc. of ACM SIGGRAPH '10*.

BARBIČ, J., AND ZHAO, Y. 2011. Real-time large-deformation substructuring. In *Proc. of ACM SIGGRAPH '11*.

BARR, A. H. 1984. Global and local deformations of solid primitives. In *Proc. of ACM SIGGRAPH '84*.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. In *Proc. of ACM SIGGRAPH '02*.

DAMBREVILLE, S., RATHI, Y., AND TANNEN, A. 2006. Shape-based approach to robust image segmentation using kernel PCA. In *Computer Vision and Pattern Recognition (CVPR) '06*.

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.

FRÖHLICH, S., AND BOTSCH, M. 2011. Example-driven deformations based on discrete shells. *Computer Graphics Forum 30*.

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

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

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

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., ZHOU, Q.-Y., VAN DE PANNE, M., COHEN-OR, D., AND NEUMANN, U. 2008. Reusable skinning templates using cage-based deformations. In *Proc. of ACM SIGGRAPH Asia '08*.

KIM, T., AND JAMES, D. L. 2009. Skipping steps in deformable simulation with online model reduction. In *Proc. of ACM SIGGRAPH '09*.

KRAUSE, R., AND WALLOTH, M. 2009. A time discretization scheme based on rothes method for dynamical contact problems with friction. *Computer Methods in Applied Mechanics and Engineering 199*, 1-4.

KRYSL, P., LALL, S., AND MARSDEN, J. E. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering 51*, 4.

LEVIN, D. I. W., LITVEN, J., JONES, G. L., SUEDA, S., AND PAI, D. K. 2011. Eulerian solid simulation with contact. In *Proc. of ACM SIGGRAPH '11*.

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*.

LI, H., WEISE, T., AND PAULY, M. 2010. Example-based facial rigging. In *Proc. of ACM SIGGRAPH '10*.

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*.

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. In *Proc. of ACM SIGGRAPH '11*.

NOCEDAL, J., AND WRIGHT, S. J. 2006. *Numerical Optimization*. Springer.

ROHMER, D., HAHMANN, S., AND CANI, M.-P. 2009. Exact volume preserving skinning with shape control. In *Proc. of Symp. on Computer Animation (SCA) '09*.

SAVOYE, Y., AND FRANCO, J.-S. 2010. CageIK: dual-laplacian cage-based inverse kinematics. In *Proceedings of the 6th international conference on Articulated motion and deformable objects (ADMO) '10*.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proc. of ACM SIGGRAPH '86*.

SEOL, Y., SEO, J., KIM, P. H., LEWIS, J. P., AND NOH, J. 2011. Artist friendly facial animation retargeting. In *Proc. of ACM SIGGRAPH Asia '11*.

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*.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. In *Proc. of ACM SIGGRAPH '05*.

WHITAKER, H., AND HALAS, J. 2002. *Timing for Animation*. Focal Press.

YAMANE, K. 2004. *Simulating and Generating Motions of Human Figures*. Springer.