Deformable Objects Alive!

Stelian Coros[1] Sebastian Martin[1] Bernhard Thomaszewski[1] Christian Schumacher[2] Robert Sumner[1]
Markus Gross[1,2]

[1]Disney Research Zurich [2]ETH Zurich

Figure 1: *Our method for controlling deformable characters allows animators to bring passive elastic objects, such as this tea cup, to life.*

Abstract

We present a method for controlling the motions of active deformable characters. As an underlying principle, we require that all motions be driven by *internal* deformations. We achieve this by dynamically adapting rest shapes in order to induce deformations that, together with environment interactions, result in purposeful and physically-plausible motions. Rest shape adaptation is a powerful concept and we show that by restricting shapes to suitable subspaces, it is possible to explicitly control the motion styles of deformable characters. Our formulation is general and can be combined with arbitrary elastic models and locomotion controllers. We demonstrate the efficiency of our method by animating curve, shell, and solid-based characters whose motion repertoires range from simple hopping to complex walking behaviors.

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

1 Introduction

Simulating deformable objects is a well-studied problem, but controlling the way they move is a completely different challenge. While most characters are articulated, many are inherently deformable and have no rigid internal structure. Examples range from simple jelly creatures to complex cartoon characters with stylized deformation behavior such as walking carpets, singing candelabra, or diligent broomsticks. Regardless of their nature, these virtual actors have to be *active*: they must walk, creep, roll or propel in some manner—and these motions need to be directable.

In order to animate articulated figures such as humanoids or animals, animators can resort to physics-based locomotion controllers that translate high-level commands (*walk left*, *crouch*, or *jump*) into sequences of joint torques that yield the desired behavior. While the problem of articulated motion control has been studied extensively, relatively little work has been done to create automatic methods that support the animation of active deformable characters.

Passive deformable objects can be easily animated using physics-based simulation, and, in general, *external* control forces can be used in order to achieve desired motion objectives. These external forces, however, typically do not sum up to zero and have non-vanishing rotational components. They can therefore change linear and angular momenta in arbitrary, non-physical ways, which may lead to unrealistic motions.

In this work, we present a new approach for intuitive control of elastically deformable characters. As an underlying principle, we allow characters to propel using only internal energy, which is generated by dynamically adapting the characters' rest shapes. Because the *internal* control forces generated by our framework are derived from an elastic potential, they automatically preserve momentum. The problem formulation we propose is aware of the forces arising from interactions with the environment, and therefore rest shape adaptations lead to desired locomotion behaviors.

Rest state adaptation is a powerful concept that provides animators with various levels of control over the resulting motion styles. In particular, we investigate two rest-pose adaptation strategies:

- *Cage-based adaptation*: the deformation of the rest state is driven by a coarse mesh that embeds the rest pose. This approach results in a very flexible parameterization of the rest shape that promotes low-frequency deformations. In addition, this strategy requires very little user input and is quick to set up.
- *Example-based adaptation*: this approach allows for artistic control over the resulting motions. In this setting, the space of rest shape deformations is restricted to an interpolation of a set of input poses, which results in a parameter space that is typically much smaller than when using cage-based adaptation.

To the best of our knowledge, our method is the first to enable simulated deformable characters to autonomously propel themselves using only internal energy. We believe that rest shape adaptation, paired with appropriate rest pose parameterizations and control laws will allow artists to easily animate a much wider class of characters than previously possible.

2 Related Work

The ways in which objects and characters move play a vital role in the creation of believable virtual worlds for video games, animated movies and training applications. The problem of creating realistic motions has been studied extensively, resulting in a wealth of literature that is relevant to our work.

Physics-based techniques provide a particularly appealing approach to animating deformable objects and characters. Starting with the pioneering work of Terzopoulos et al. [1987], many research projects have been aimed at increasing the diversity, efficiency, and accuracy of physical simulations. Since space does not permit us to review a representative set of related works, we only mention the two methods which found direct application in our framework: the discrete shell model by Grinspun et al. [2003] for deformable surfaces and the finite element model by Irving and colleagues [2007] for volumetric solids.

Simulating physical objects efficiently is only half the battle however. A great deal of control over the resulting motions is also required in order to provide artists with the power needed to create compelling visual scenes. At the same time, it is very important that the resulting motions remain physically plausible. Despite receiving a great deal of attention from the research community, this remains to a large extent an open problem [O'Brien 2011]. We aim to address this challenge, and to help position our work with respect to previous methods, we group existing approaches as follows:

Control of passive objects: It is oftentimes desirable to simulate seemingly passive objects whose behaviors are, in fact, user-controlled. To this end, Twigg and James [2007] present a method that exploits the speed of multi-body simulations. By simulating numerous variations of the same scene in parallel, the proposed system lets users interactively chose desirable outcomes and therefore control the motions of the simulated objects. The system proposed by Popović et al. [2000] automatically manipulates various simulation parameters, such as masses, moments of inertia, surface normals and elastic coefficients in order to allow simple dynamically-simulated objects to achieve user-set goals. Twigg and Kačić-Alesić [2011] perform off-line optimizations of the rest lengths of springs in a deformable object simulation in order to maintain the look of artist-modeled objects once gravity is applied to a scene. Martin et al. [2011] describe a framework that allows users to steer, through examples, the behavior of passive deformable objects as they interact with their environments. In contrast to this line of research, we are interested in animating deformable objects as if they were alive, autonomous characters.

Control of soft objects using external forces: A popular approach to controlling the behavior of deformable objects is through the use of external forces. Barbič and Popović [2008] use a time-varying Linear Quadratic Regulator controller to compute forces that steer simulated deformable objects or fluids towards desired target configurations, and Jeon and Choi [2007] use a combinatorial discrete optimization approach to obtain physically-plausible, user-controlled motions for cloth. Barbič et al. [2009] present a space-time optimization method that operates in a low-dimensional subspace defined by input keyframes. The optimization aims to eliminate errors in tracking the input keyframes while minimizing errors in the dynamics. These errors can be interpreted as the influence of applied external forces. This method was applied to FEM and mass-spring simulations of deformable objects. TRACKS [Bergou et al. 2007] enforces the low-frequency motions of thin shells using constrained Lagrangian mechanics, while allowing high-frequency motions to naturally appear due to the object's dynamics. The adjoint method, a non-linear gradient-based optimization approach, was successfully applied to control the behavior of fluid simulations [McNamara et al. 2004] and particle systems [Wojtan et al. 2006]. Our work shares many of the goals of these existing methods. However, instead of using non-conservative external forces, the motions we generate are the result of an internal potential field that is automatically modulated.

Control of articulated rigid figures: The concept of using forward dynamics simulations in conjunction with control policies for the purpose of character animation was introduced over two decades ago [Girard and Maciejewski 1985; Raibert and Hodgins 1991]. Since then, a large number of control frameworks applicable to characters represented by articulated rigid body hierarchies have been presented. Examples include, among others, control methods for legged locomotion [Yin et al. 2007; Wu and Popović 2010; Coros et al. 2010; Lee et al. 2010; de Lasa et al. 2010] (and many others), athletic activities [Hodgins et al. 1995], swimming [Tu and Terzopoulos 1994; Tan et al. 2011] and flying [Wu and Popović 2003]. These control policies generate joint torques that, when applied to the joints of the articulated figures, result in purposeful motions. This approach has been shown to generalize well to characters that are simulated using two-way coupling between articulated rigid bodies and deformable objects [Jain and Liu 2011; Kim and Pollard 2011]. The joint torques applied to the articulated figures are used to actively drive the motions, while the deformable objects are simulated passively. In contrast, in this paper we investigate the problem of generating desirable internal forces for continuously deforming characters that have no well-defined internal or external rigid structure. In this setting, concepts such as torques and joints do not exist. We therefore use a different, more general approach to motion control that works by automatically adjusting the objects' rest configurations.

Control of soft objects using internal forces: Real creatures move around their environments using only internal energy, which is generated through muscle contractions. Simulating muscles and their effects on the motions of virtual characters is a topic that has received much attention from the academic community. This line of research holds the promise of creating virtual characters whose motions are indistinguishable from the motions of real humans and animals. While this goal is not yet within reach, significant progress has been made. For instance, Sueda et al. [2008] introduced a controllable strand-based muscle model that was used to create realistic animations of a human hand. Teran et al. [2005] used finite elements to simulate muscles of a virtual human torso whose motions were kinematically controlled. Tu et al. [1994] modeled fish as mass spring systems, and their motions were controlled by adjusting the springs' rest lengths. Here we extend the concept of muscles to arbitrary deformable objects. Our approach is related to the work of Ijiri et al. [2009], who introduced a system that lets users manually specify deformation fields for the rest poses of solid deformable objects. In contrast to this work, our method automatically computes appropriate rest pose configurations that allow simulated rods, shells and solids to walk, crawl and hop autonomously.

3 Method Overview

In the framework we propose, rest state adaptations provide the means of actuation and propulsion for deformable characters. The goal of our method is therefore to automatically compute changes to the rest configuration of the objects in a way that results in directed

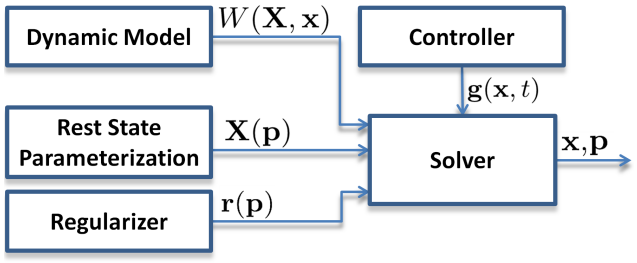

Figure 2: *Overview of our method: Our solver takes as input a dynamic model which describes an object's internal energy, a set of objectives, a parameterization of the rest pose and a regularizer. The output consists of an appropriate rest pose and the corresponding state of the dynamic model at the next time step.*

motion. In this section we describe the essence of our approach and its different components, which are also illustrated in Fig. 2.

Dynamic Models We start directly in the discrete setting and let $\mathbf{X}$ and $\mathbf{x}$ denote vertex positions of a deformable object (curve, shell or solid) in the undeformed and deformed configurations respectively. The deformation behavior of the object is governed by an elastic potential $W(\mathbf{X},\mathbf{x})$, which gives rise to internal forces $\mathbf{f}_{\text{int}} = -\partial W/\partial \mathbf{x}$. Together with external forces $\mathbf{f}_{\text{ext}}$ (gravity, contacts, friction), the dynamic behavior is described by the well-known equations of motion

$$\mathbf{M}\ddot{\mathbf{x}} - \mathbf{f}_{\text{int}}(\mathbf{X},\mathbf{x}) - \mathbf{f}_{\text{ext}} = \mathbf{0}\,, \tag{1}$$

where $\mathbf{M}$ is the mass matrix and $\ddot{\mathbf{x}}$ are nodal accelerations. For the purpose of motion control, we dismiss the option of adding external forces that are not the result of correct physical interactions with the environment. Instead, we require that all motions be a direct result of internal deformations. To this end, we consider $\mathbf{X}$ as the sole *control* parameters. In other words, we adapt the rest shape of the object to yield the required deformations for motion control. This approach is similar in spirit to the way muscles work and, as for muscles, the resulting control forces are always momentum conserving.

In order to formalize this approach, let $\mathbf{g}(\mathbf{x},t)$ denote a vector-valued objective function whose components measure the deviation from a given set of (to be defined) time-dependent motion goals. We seek to find undeformed and deformed configurations that minimize the distance to the motion objectives while satisfying the equations of motions (1). This can be cast as a constrained optimization problem of the form

$$\min_{\mathbf{x},\mathbf{X}} \frac{1}{2}\left|\mathbf{g}(\mathbf{x},t)\right|^2 \quad s.t. \quad \mathbf{c}(\mathbf{X},\mathbf{x},\ddot{\mathbf{x}}) = 0\,, \tag{2}$$

where we use $\mathbf{c}$ to encode (1) as a hard constraint. In summary, we seek to find optimal control parameters $\mathbf{X}$ that best achieve the motion objectives $\mathbf{g}$ while exactly satisfying the physics constraints $\mathbf{c}$. We solve this problem numerically using Lagrangian optimization but postpone details to Sec. 6.

The optimization problem above can be combined with any time integration scheme. Using the well-known implicit Euler method, Equation (1) is discretized in time as

$$\mathbf{M}\left(\frac{\mathbf{x}_{n+1}}{h^2} - \frac{\mathbf{x}_n}{h^2} - \frac{\mathbf{v}_n}{h}\right) - \mathbf{f}_{\text{int}}(\mathbf{X},\mathbf{x}_{n+1}) - \mathbf{f}_{\text{ext}} = 0\,, \tag{3}$$

for a given time step n. In this form, the hard constraints can be interpreted as an implicit map $\mathbf{X} \mapsto \mathbf{x}_{n+1}$ that parameterizes the space of possible solutions. From this space, our solver chooses the solution that minimizes the objective function.

Rest State Parametrization In the most general setting, every vertex of the rest shape is an independent control variable for the optimization problem described by Equation (2). For deformable objects with a large number of vertices, this can lead to optimization problems with an unwieldy number of parameters. To address this, we show that using reduced linear subspaces for rest pose deformations is a simple, yet very powerful strategy. In particular, we choose the rest state configuration $\mathbf{X}(\mathbf{p})$ as

$$\mathbf{X}(\mathbf{p}) = \mathbf{X}_0 + \mathbf{L}\mathbf{p}, \tag{4}$$

where $\mathbf{X}_0$ is the initial rest state, $\mathbf{p}$ describes generalized displacements, and $\mathbf{L}$ is a linear map between changes in rest state coordinates $\mathbf{X}$ and the reduced control variables $\mathbf{p}$. With this representation, we can restate our optimization problem in terms of the state variables $\mathbf{x}$ and the reduced control parameters $\mathbf{p}$ as

$$\min_{\mathbf{x},\mathbf{p}} \frac{1}{2}\left|\mathbf{g}(\mathbf{x},t)\right|^2 + \mathbf{r}(\mathbf{p})\; s.t.\; \mathbf{c}(\mathbf{X}(\mathbf{p}),\mathbf{x}) = 0\,, \tag{5}$$

where it is implied that, as per Equation (3), nodal accelerations $\ddot{\mathbf{x}}$ are an explicit function of $\mathbf{x}$. The additional regularizing term $\mathbf{r}(\mathbf{p})$ will be described shortly. The linear map $\mathbf{L}$ determines the space of allowable rest shapes, and implicitly, the capabilities of the deformable objects we seek to control. Selecting appropriate linear maps is, therefore, of great importance for our framework. We discuss two practical approaches to choosing flexible and efficient rest pose parameterizations in Sections 4 and 5.

Contacts and Friction We implement contact and friction forces using implicit springs that are active only when vertices of the simulation mesh are in contact with external objects. For each colliding (deformable) vertex i, we first compute the point of contact $\mathbf{c}_i$ by projecting onto the external triangles. We then apply a contact force that is proportional to deviations from $\mathbf{c}_i$ in the triangle's normal direction and a friction force that is proportional to in-plane deviations from $\mathbf{c}_i$. The contact and friction forces are added to the $\mathbf{f}_{\text{ext}}$ term in equation (3), which then becomes an explicit function of $\mathbf{x}_{n+1}$.

Adapting the rest shape affects the motion of all vertices and thus the forces generated at the contact points. The implicit nature of the friction and contact forces makes them *visible* to our optimization method, which automatically exploits them in order to help the characters propel.

Controller Having established the basic means for deformable object actuation, we now describe a simple controller that translates high-level motion goals (*walk left, hop, etc.*) into sequences of objectives that can be cast into algebraic functions of the state variables via $\mathbf{g}(\mathbf{x},t)$. This vector-valued objective function is the result of concatenating an arbitrary number of constraints that are linear in the positions of the vertices:

$$\mathbf{g}(\mathbf{x},t) = [w_1\mathbf{g}_1(\mathbf{x},t), w_2\mathbf{g}_2(\mathbf{x},t), ..., w_n\mathbf{g}_n(\mathbf{x},t)]^T, \tag{6}$$

$$\mathbf{g}_i(\mathbf{x},t) = \sum_j w_{xj}\mathbf{x}_j - s_i(t), \tag{7}$$

where $s(t)$ indicates a time varying goal target. The weight w_i is used to scale the importance of goal $\mathbf{g}_i$ relative to the other goals. Each goal constraint $\mathbf{g}_i$ affects a subset of the object's vertices, as

indicated by the weights w_{xj}. As an example, for the center-of-mass constraints we use throughout the paper, the vertex weights w_{xj} are chosen as $m_j / \sum_i m_i$. For the vertices not affected by a specific goal, we simply set the weights w_{xj} to zero.

It is often convenient to represent constraints as a function of velocity, rather than positions. For instance, to get an object to hop, a controller can simply output a desired upwards velocity for the center of mass. The goal targets are therefore exposed as:

$$\hat{\mathbf{g}}_i(\mathbf{x}, t) = k_p(\sum_j w_{xj}\mathbf{x}_j - s_{xi}(t)) + k_d(\sum_j w_{xj}\mathbf{v}_j - s_{vi}(t)) \quad (8)$$

where k_p, k_d, $s_{xi}(t)$, and $s_{vi}(t)$ are provided by the controller. By expressing the velocity of each vertex as a function of the positions at the previous time step $\mathbf{x}_n$, $\mathbf{v} = (\mathbf{x} - \mathbf{x}_n)/h$, we represent the goal target from Equation (7) as:

$$s_i(t) = \frac{h}{hk_p + k_d}(k_p s_{xi} + k_d s_{vi} + \frac{k_d}{h}\sum_j w_{xj}\mathbf{x}_{nj}) \quad (9)$$

and the goal weight from Equation (6) as:

$$w_i(t) = \frac{hk_p + k_d}{h} \quad (10)$$

We note that solving Equation (5) is equivalent to simultaneously solving for the internal control forces that minimize the objective $\mathbf{g}$ and integrating the system state forward in time. As a result, the effective planning horizon of our method is equal to the simulation time step. Nevertheless, this simple control strategy is both effective and intuitive. As described in more detail in Section 7, we used it to create hopping and rolling motions for deformable objects of various shapes and sizes. We also used it successfully to control more complicated behaviors such as bipedal walking. For these motions, control strategies that output high-level motion features such as targets for the positions of the center of mass and feet [Coros et al. 2010; de Lasa et al. 2010] are a natural fit for our controller.

Regularizer A rest shape which is optimal with respect to Equation (2) is not necessarily desirable in all regards. For example, we generally prefer smoothly deformed rest shapes over ones with large isolated distortions. Such preferences can be expressed formally in terms of a regularizing potential $\mathbf{r}(\mathbf{p})$ that is added to the objective term as in Equation (5). The concrete form of the regularizer depends on the rest pose parametrization and we postpone details to Sections 4 and 5.

4 Subspaces for Rest Shape Adaptations

The previous section has laid out the basis for controlling deformable objects by adapting their rest shapes. As we will show, the way in which this adaptation occurs greatly impacts the resulting motion. Conceptually the simplest case is to consider each vertex of the rest shapes as a control parameter in the optimization problem (5). For this case, the linear map $\mathbf{L}$ from Equation (4) is an identity matrix of appropriate dimensions. Fig. 3 (*a*) demonstrates this approach on a simple example in which an *I*-shaped deformable character is asked to hop to the left. While this adaptation strategy can reveal unexpected but plausible means of locomotion, it also has disadvantages. First, it results in larger problems, which are more difficult and expensive to solve. Second, it can potentially lead to large, non-smooth changes of the rest shape.

Regularizer Undesirable distortions can be counteracted by introducing an adequate regularizer. Visually, if two solutions are similar in terms of Equation (5) we prefer the one that is *closer* to the initial rest shape $\mathbf{X}_0$. This can be formalized by introducing a regularizer based on the elastic potential between the current and initial rest poses:

$$\mathbf{r}(\mathbf{p}) = W_{\text{reg}}(\mathbf{X}_0, \mathbf{X}(\mathbf{p})) \,, \quad (11)$$

where we choose the regularizing potential W_{reg} to be of the same form as the elastic potential W but allow for different material parameters. This is convenient if, for instance, changes in volume should be penalized more severely than other deformations.

Coarse-Scale Adaptation It is often not necessary or even desirable to leverage all degrees of freedom in order to create purposeful motions. For example, having the center of mass of a character follow a given trajectory can often be achieved using only low-frequency deformations.

A common approach for applying coarse-scale deformations to high-resolution geometry are spatial deformation methods. Harmonic coordinates [Joshi et al. 2007] are particularly attractive for our setting: the deformation field is defined by the vertices of an enclosing polygonal cage (no vertices on the inside), it is linear with respect to the cage vertices, and it is smooth (no isolated distortions or intersections). More concretely, let $\mathbf{C}_j \in \mathbb{R}^3$ denote the cage vertices and let h_j be the associated harmonic coordinate function. The deformed position of a given rest state vertex $\mathbf{X}_{d,i} \in \mathbb{R}^3$ is then defined as

$$\mathbf{X}_{d,i} = \mathbf{X}_{0,i} + \sum_j h_j(\mathbf{X}_{0,i})\mathbf{C}_j = \mathbf{X}_{0,i} + \sum_j h_{ij}\mathbf{C}_j \,.$$

The harmonic coordinates h_{ij} are directly identified with the entries of the linear map in (4) as $\mathbf{L}_{ij} = h_{ij}$, and the control parameters $\mathbf{p}$ become the cage vertices $\mathbf{C}_j$. The coordinates have to be computed numerically but since they are defined over the initial rest state $\mathbf{X}_0$, these computations have to be done only once.

Fig. 3 (*b*) shows the result of the cage-based adaptation scheme applied to the *I*-character. It can be noted that the deformations of the rest state are much smoother than for the case of unrestricted rest shape adaptation. This effect translates directly into smoother deformed shapes and, thus, a notably different style of motion.

The cage-based scheme introduced here allows us to restrict rest shape adaptations to lower frequency deformations. This significantly reduces the search space, thus making the optimization problem easier to solve. However, this method still lacks direct control over the type of deformations the rest shape may undergo. We address this by resorting to example-based deformation subspaces, as described next.

5 Example-Based Adaptation

Being able to exert control over the rest shape adaptation is an important feature since the ways in which a character moves determine its personality. In order to further structure the space such as to reflect a characteristic style of motion, we turn towards *example-based* techniques.

The concept of example-based rest shape control can be summarized as follows: we let the artist choose a set of example poses that describe desirable character deformations; we then convert these poses into a representation suitable for interpolation and use convex combinations of the examples as the subspace for rest shape adaptations.

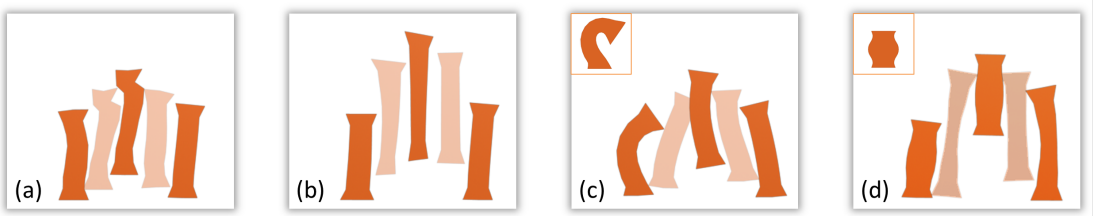

Figure 3: *Different rest state adaptation strategies lead to different locomotion styles: unrestricted* (a)*, cage-based* (b)*, example-based with a bent* (c) *and a compressed* (d) *input pose (shown as inset figures in the upper left corners).*

A central question in this context is how to best interpolate between shapes. While the number of existing methods is abundant, most of them follow a common scheme: the input shapes are first converted into an intermediate representation such as Laplacian coordinates, deformation gradients or Green strains. The interpolated data is then obtained as a weighted (convex) combination of the input data. Finally, interpolated geometry is reconstructed by solving a system of (nonlinear) equations. From a practical point of view, having an implicit and possibly nonlinear relation between the input and interpolated rest shapes can make the computation of rest state derivatives that are required to solve Equation (5) quite complex. Ideally, we would like to avoid the reconstruction step altogether and directly work on interpolated representations. Fortunately, this can be achieved in a simple and efficient manner as described next.

5.1 Incompatible Shape Interpolation

We start by pointing out two important facts: first, discrete elastic energies are usually computed by summing up elemental contributions. Second, these elemental contributions are typically based on rotation-invariant deformation measures such as the Green strain. As a consequence, cutting the rest state of a given deformed object into disjoint elements and applying a random rotation to each of them does not change the elastic energy or its derivatives.

Motivated by these observations, we resort to an incompatible discretization and model the rest state as an assembly of disjoint elements. Similar approaches have been pursued before, albeit for different purposes such as geometric modeling [Botsch et al. 2006] or remeshing [Wicke et al. 2010]. Compared to the compatible setting, it has the striking advantage that *shape* interpolation reduces to *element* interpolation which is much simpler: we first register each element in its different poses to a common reference frame by factoring out the rotations of the individual poses. In this common frame, interpolating between different element poses is then simply a matter of interpolating vertex positions. Using this interpolation scheme for example-based rest shape adaptations, we can compute energies and their rest state derivatives in a simple and efficient way.

In order to translate this concept to the formal setting of Equation (4), we let $\mathbf{X}_0^e$ denote the vector of concatenated vertex positions $\mathbf{X}_{0,i}^e \in \mathbb{R}^3$, $i = \{1, \dots, 4\}$, for a given undeformed tetrahedral element e and encode all its example poses as corresponding vectors $\mathbf{X}_k^e$. We then compute the deformation gradient $\mathbf{F}_k^e = \mathbf{S}_k \mathbf{S}_0^{-1}$ between the undeformed element and each example, where $\mathbf{S}_k, \mathbf{S}_0 \in \mathbb{R}^{3\times3}$ are matrices with columns $\mathbf{S}_{k,j} = \mathbf{X}_{k,j} - \mathbf{X}_{k,4}$, with $j = \{1, 2, 3\}$, holding the elements' edge vectors. Finally, we apply a polar decomposition to $\mathbf{F}_k^e$ to obtain the rotation $\mathbf{R}_k^e$ that aligns example k with the reference frame as $\hat{\mathbf{X}}_{k,i}^e = (\mathbf{R}_k^e)^T \mathbf{X}_{k,i}^e$. The concatenations of these elemental vectors then defines an interpolated rest shape as

$$\mathbf{X}(\mathbf{p}) = \mathbf{X}_0 + \sum_k (\hat{\mathbf{X}}_k - \mathbf{X}_0)\mathbf{p}_k \,. \tag{12}$$

This incompatible rest state interpolation is very efficient since the unrotated positions $\hat{\mathbf{X}}_j^e$ stay constant over time and can thus be precomputed. Equation (12) can be cast into the form of (4) by choosing the entries of the linear operator as $\mathbf{L}_{ij} = (\hat{\mathbf{X}}_j - \mathbf{X}_0)_i$. It should be noted that, compared to unrestricted and cage-based adaptation, the dimension of $\mathbf{X}(\mathbf{p})$ is higher since vertices are no longer shared among the elements. Computation times are, however, largely governed by the size of $\mathbf{p}$, which is now determined by the number of example poses. Since the latter is typically much smaller than the number of control parameters used for unrestricted or cage-based adaptation, the example-based approach is more efficient in this regard as well.

Using examples to define a subspace for rest shape adaptation provides an animator with explicit control over the styles of the resulting motions. Fig. 3 (*c*) and (*d*) illustrate this technique applied to the *I*-character. The example space is defined by the initial undeformed pose and one additional example, a pose where the *I* is bent (*c*), respectively, compressed (*d*). In both cases, it can be noticed that the character makes heavy use of the example pose to gather enough energy for the jump. The different example poses lead to substantially different yet plausible jumping styles that respect the artistic intent.

Regularizer Although example-based interpolation behaves robustly for moderate extrapolation, the behavior in this regions is typically less intuitive and we therefore require the rest shape to remain within the convex hull of the examples. In the case of global examples that extend over the entire domain of the object, this is achieved by introducing a regularizing term as $\mathbf{r}(\mathbf{p}) = \frac{1}{2}(\sum_i \mathbf{p}_i - 1)^2$. Similarly, negative values of the control parameters are penalized by the regularizer.

It is sometimes more convenient to define example poses that only affect localized regions of a deformable object. This is the case, for instance, when specifying poses that control the range of possible deformations of individual legs. We treat these cases by setting up convexity constraints on the corresponding subsets of the control parameters.

5.2 Extension to Curves and Shells

The interpolation described by Equation (12) applies equally for both 2D and 3D finite elements. For the case of elastic curves and shells, we additionally have to account for bending deformation. We will only describe the approach for shells here since the formulation for curves is completely analogous. Following Grinspun et al. [2003], we measure the difference between a discrete undeformed triangle mesh and its deformed counterpart in terms of differences in edge lengths and dihedral angles. Recent work [Winkler et al. 2010; Fröhlich and Botsch 2011] already demonstrated that linear interpolation in this length-angle strain space leads to appreciable and intuitive behavior.

In analogy to the 3D case, we resort to an incompatible rest state and represent a shell surface with n_e edges by a set of n_e disjoint hinge elements, each consisting of two edge-adjacent triangles. We first register the different example poses of each hinge element to a common frame by applying affine transformations that align the center edges with their reference counterpart. This setting allows us to linearly interpolate vertex positions between hinge elements and then extract lengths and angles from the interpolated geometry. Note that for stretch deformation, doing so is exactly equivalent to interpolating edge lengths between the disjoint hinge elements. For bending deformation, this approach leads to a different interpolation speed compared to angle interpolation but is otherwise equivalent. As for the 3D case, the central advantage of this incompatible approach is that the element-wise reconstruction is analytical whereas the compatible setting requires a (nonlinear) reconstruction step.

5.3 Validation

We have used example-based rest shape adaptations for several of the animations presented in Sec. 7 and observed desirable behavior both in terms of directability and numerical performance. In order to better isolate the behavior of our incompatible interpolation scheme, we additionally compare with two reference solutions: the method by Martin et al. [2011] for solids and the one by Fröhlich and Botsch [2011] for shells. Both use a strain space representation for interpolating between examples poses: Green strain for solids, edge lengths and angles for shells. Note that, while each geometric configuration has its strain space counterpart, the reverse is not true, which is why both methods perform a closest point projection in a least squares sense. Similarly, our incompatible configurations do not generally have exact compatible analogues. For comparison, we reconstruct compatible meshes by solving a static problem with the interpolated incompatible mesh as rest state.

For solids, we interpolate between two poses of a bar, one straight the other twisted, using 100 uniformly-sampled weights $\alpha_i \in [0, 1]$ and reconstruct the corresponding geometries. Our results match the reference solution very closely for all α values. Since differences in geometry are hardly perceptible, we only show interpolation results obtained with our method in Fig. 4.

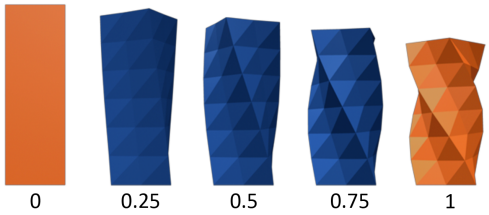

Figure 4: *Interpolation between two poses of a solid bar (orange) using our incompatible scheme (blue).*

To test the incompatible interpolation scheme for shells, we applied our method to one of the examples used by Fröhlich and Botsch [2011], consisting of a straight and a helically-twisted tube. Fig. 5 shows the results of our method compared to the reference solution for three sample weights. We obtained good overall shape approximation without any stretching artifacts, which can be attributed to the constant-rate interpolation of the edge lengths. Note that, as expected, the geometries do not match exactly since our interpolation speed for angles is different from the reference solution.

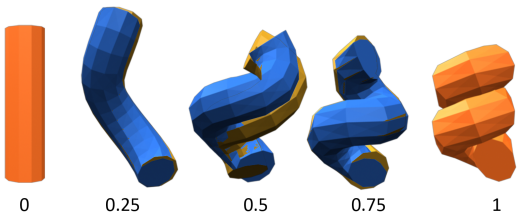

Figure 5: *Interpolation between two poses of a shell (orange) using our method (blue) and a reference implementation (yellow).*

6 Lagrangian Optimization

The motion of our controlled deformable characters is governed by the optimization problem (5), which is nonlinear in both objective function and constraints. In order to solve Equation (5) in every time step, we employ Sequential Quadratic Programming (SQP) [Nocedal and Wright 2000] and solve the arising linear systems with a custom-tailored block solver.

SQP Formulation Following the SQP algorithm, we start by defining a Lagrangian function for the optimization problem as

$$L(\mathbf{x}, \mathbf{p}, \lambda) = \frac{1}{2} |\mathbf{g}(\mathbf{x}, t)|^2 + \mathbf{r}(\mathbf{p}) + \lambda^T \mathbf{c}(\mathbf{X}(\mathbf{p}), \mathbf{x}) \, .$$

Using a Newton-Raphson method, we iteratively solve the first order optimality conditions

$$\begin{aligned} 0 &= \frac{\partial L}{\partial \mathbf{x}} = \mathbf{g}(\mathbf{x}, t) \frac{\partial \mathbf{g}}{\partial \mathbf{x}} + \lambda^T \left(\frac{1}{h^2} \mathbf{M} - \frac{\partial \mathbf{f}_{\text{tot}}}{\partial \mathbf{x}} (\mathbf{X}(\mathbf{p}), \mathbf{x}) \right) \\ 0 &= \frac{\partial L}{\partial \mathbf{p}} = \frac{\partial \mathbf{r}}{\partial \mathbf{p}} (\mathbf{p}) - \lambda^T \frac{\partial \mathbf{f}_{\text{int}}}{\partial \mathbf{X}} (\mathbf{X}(\mathbf{p}), \mathbf{x}) \frac{\partial \mathbf{X}}{\partial \mathbf{p}} \\ 0 &= \frac{\partial L}{\partial \lambda} = \mathbf{M} \ddot{\mathbf{x}} - \mathbf{f}_{\text{tot}}(\mathbf{X}(\mathbf{p}), \mathbf{x}) \end{aligned}$$

where we used $\mathbf{f}_{\text{tot}}(\mathbf{X}(\mathbf{p}), \mathbf{x}) = \mathbf{f}_{\text{int}}(\mathbf{X}(\mathbf{p}), \mathbf{x}) + \mathbf{f}_{\text{ext}}(\mathbf{x})$ to account for the fact that $\mathbf{f}_{\text{ext}}$ includes position-dependent contact forces. In each iteration, we compute corrections $\Delta\mathbf{x}$, $\Delta\mathbf{p}$ and $\Delta\lambda$ by solving a linear system of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{D}^T \\ \mathbf{B} & \mathbf{C} & \mathbf{E}^T \\ \mathbf{D} & \mathbf{E} & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{p} \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} , \qquad (13)$$

whose matrix blocks $\mathbf{A} = \partial_{\mathbf{xx}} L$, $\mathbf{B} = \partial_{\mathbf{px}} L$, $\mathbf{C} = \partial_{\mathbf{pp}} L$, $\mathbf{D} = \partial_{\lambda\mathbf{x}} L$ and $\mathbf{E} = \partial_{\lambda\mathbf{p}} L$ correspond to the partial Hessians of the Lagrangian and the vectors $\mathbf{a} = \partial L / \partial \mathbf{x}$, $\mathbf{b} = \partial L / \partial \mathbf{p}$ and $\mathbf{c} = \partial L / \partial \lambda$ are the partial gradients of L. Having solved for the search direction, we determine an admissible step size using a linesearch method as

$$\begin{bmatrix} \mathbf{x}_{n+1} \\ \mathbf{p}_{n+1} \\ \lambda_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_n \\ \mathbf{p}_n \\ \lambda_n \end{bmatrix} - \alpha \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{p} \\ \Delta\lambda \end{bmatrix} . \qquad (14)$$

Starting with a full step of $\alpha = 1.0$, we successively half the step size if the trial solution increases the value of the merit function

$$\psi(\mathbf{x}, \mathbf{p}) = \frac{1}{2} |\mathbf{g}(\mathbf{x}, t)|^2 + \mathbf{r}(\mathbf{p}) + \gamma \, |\mathbf{c}(\mathbf{X}, \mathbf{x})|_{L^1} \, .$$

The parameter γ allows us to adaptively control the importance of constraint satisfaction. We start with a small value of γ to allow for larger steps in the beginning and then progressively increase its value to obtain accurate constraint satisfaction upon convergence.

Block Solver The linear system (13) is symmetric, indefinite, and has a special block structure: $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{E}$ are dense, whereas $\mathbf{A}$ is sparse. While there are efficient direct solvers for symmetric indefinite problems, the dense blocks significantly deteriorate the performance of factorization methods.

Fortunately, we can solve (13) efficiently with a factorization method by using Gaussian block elimination. This step takes advantage of the fact that $\mathbf{D} = 1/h^2\mathbf{M} - \partial(\mathbf{f}_{\text{tot}})/\partial\mathbf{x}$ is invertible. From the last row of (13) we deduce $\Delta\mathbf{x} = \mathbf{D}^{-1}(\mathbf{c} - \mathbf{E}\Delta\mathbf{p})$ and, rearranging terms, obtain

$$\begin{aligned}(\mathbf{B}^T - \mathbf{AF})\Delta\mathbf{p} + \mathbf{D}^T\Delta\lambda &= \mathbf{a} - \mathbf{Ad} \\ (\mathbf{C} - \mathbf{BF})\Delta\mathbf{p} + \mathbf{E}^T\Delta\lambda &= \mathbf{b} - \mathbf{Bd}\,,\end{aligned}$$

where $\mathbf{F} = \mathbf{D}^{-1}\mathbf{E}$ and $\mathbf{d} = \mathbf{D}^{-1}\mathbf{c}$. We solve the first equation symbolically for $\Delta\lambda$ and substitute the resulting expression into the second equation to obtain a linear system for $\Delta\mathbf{p}$ as

$$\mathbf{K}\Delta\mathbf{p} = \mathbf{b} - \mathbf{Bd} - \mathbf{F}^T\mathbf{a} + \mathbf{F}^T\mathbf{Ad}\,, \tag{15}$$

where $\mathbf{K} = (\mathbf{C} - \mathbf{BF} - \mathbf{F}^T\mathbf{B}^T + \mathbf{F}^T\mathbf{AF})$. Having precomputed $\mathbf{F}$ and $\mathbf{d}$, we numerically solve for $\Delta\mathbf{p}$ and use the result to successively compute the remaining corrections

$$\mathbf{D}^T\Delta\lambda = \mathbf{a} - \mathbf{Ad} + (\mathbf{AF} - \mathbf{B}^T)\Delta\mathbf{p} \tag{16}$$

$$\mathbf{x} = \mathbf{d} - \mathbf{F}\Delta\mathbf{p}\,. \tag{17}$$

Note that this block-based formulation can be implemented very efficiently using a sparse factorization solver: prefactoring the (symmetric) matrix $\mathbf{D}$ once, we can compute $\mathbf{F}$ and $\mathbf{d}$ using backsubstitution and solve (16) in the same manner. The linear system (15) is dense, but since it is only of size $dim(\mathbf{p})$ the computation costs are insignificant compared to the sparse solve for $\mathbf{D}$.

7 Results

We demonstrate the versatility of our approach by creating crawling, rolling, hopping and walking motions for various deformable objects. Our results, which are best seen in the accompanying video, are summarized in Table 1, and snapshots of some of the objects we controlled are shown in Fig. 6. We apply our method to 2D curves and solids, as well as thin shells and volumetric solids in 3D. The elastic potentials that we used for the various deformable models are described in Appendix A.

Control The simplest approach to controlling the behavior of deformable objects is to prescribe desired values for the center of mass position or velocity. Note, however, that our optimization framework cannot guarantee that these objectives are always well-satisfied. For instance, if an object is in mid-air, changing the rest pose configuration has no effect on the position of the center of mass or its rate of change. Indeed, our characters can only propel by purposefully interacting with their environment through friction and contact forces—a strategy that is automatically discovered. The *Tea Cups*, *3D Cross*, *Ace* and letters *S*, *G*, *I* and *P* were controlled using this strategy.

Center-of-mass control is well suited to rolling and hoping motions. For more difficult control problems, however, this strategy starts to become ineffective. For instance, ensuring that a character lands on its feet after a hop requires specific initial conditions for the take-off, which may not be sufficiently well represented by the objectives placed on the center of mass. This is also the case for the walking behaviors we implemented. For these, we add additional soft constraints that act on vertices on the feet of the characters in order to control foot placements. Similarly, constraints on vertices on the body are used to control the character's orientation. The targets for the positions of the feet and the center of mass are computed using a strategy similar to the one described by Coros et al. [2010]. The use of foot-placement feedback is particularly important to ensure that characters do not lose their balance. We applied the walking controllers to the letters *A*, *R* and *H*, the *Castle*, *Pirate* and *Cheesy* characters.

Rest Pose Parameterization The cage rest pose parameterization that we describe provides a great deal of flexibility (depending on the resolution of the cage, of course), making it useful for all the types of controllers we implemented. However, it does not allow users to explicitly affect the deformations that the characters undergo. In the spirit of [Martin et al. 2011], we address this limitation by allowing the space of rest-pose deformations to be explicitly specified through examples. A static solver or mesh editing software was used to create the inputs.

For the *3D Cross* character shown in Fig. 6 we used one example for each leg, which allowed for stretching motions. For *Pirate*, we provided groups of example poses that locally deformed the feet, hips and the length of each leg. These poses were designed to allow the character to control its center of mass and foot placements. We also used examples to animate *Ace*, a 3D shell character.

The *Crusty* characters (Fig. 6) were created to illustrate the difference between cage-based and example-based rest shape adaptation. The dark *Crusty* character was controlled using the cage-based strategy. The side-to-side motion emerges naturally. The second *Crusty* character is controlled using the example-based strategy. Through the example rest poses we provided, the side-to-side motion was removed, and instead, a desirable slight twist about the vertical axis was introduced. The kinematic objectives were identical for the two tests.

Timing Information The results we described here run in real-time or at interactive rates. The average time spent computing one second of simulated motion for each example is given in Table 1. All simulations were performed with a step size between 0.01s and 0.033s, on a single core of an Intel Core i7 960, 3.2 GHz. For visualization purposes, we used high-resolution meshes embedded in relatively low-resolution simulation meshes.

Model	#x	#p	RPP	DM	time (s)
Sphere	246	24	cage	3D solid	2.6
Ace 1	231	3	examples	3D shell	9.2
Ace 2	231	42	cage	3D shell	14.1
Castle 1	142	22	cage	2D solid	1.1
Castle 2	64	22	cage	2D rod	0.57
Tea Cups	228-375	24	cage	3D solid	5.64-16.5
S and G	100-130	8-36	cage	2D solid	0.8-2.1
I and P	54-64	3	ex	2D solid	0.8-0.9
A, R, H	66-104	20-28	cage	2D solid	0.8-1.9
3D Cross	1128	6	ex	3D solid	21.7
Pirate	188	12	ex	2D solid	3.16
Cheesy	243	150	cage	3D solid	8.6
Crusty 1	114	12	ex	3D solid	7.8
Crusty 2	114	57	cage	3D solid	9.8

Table 1: *Summary of results with columns indicating the number of DOFs, number of control parameters, adaptation scheme, elastic model and average computation time for one second of animation.*

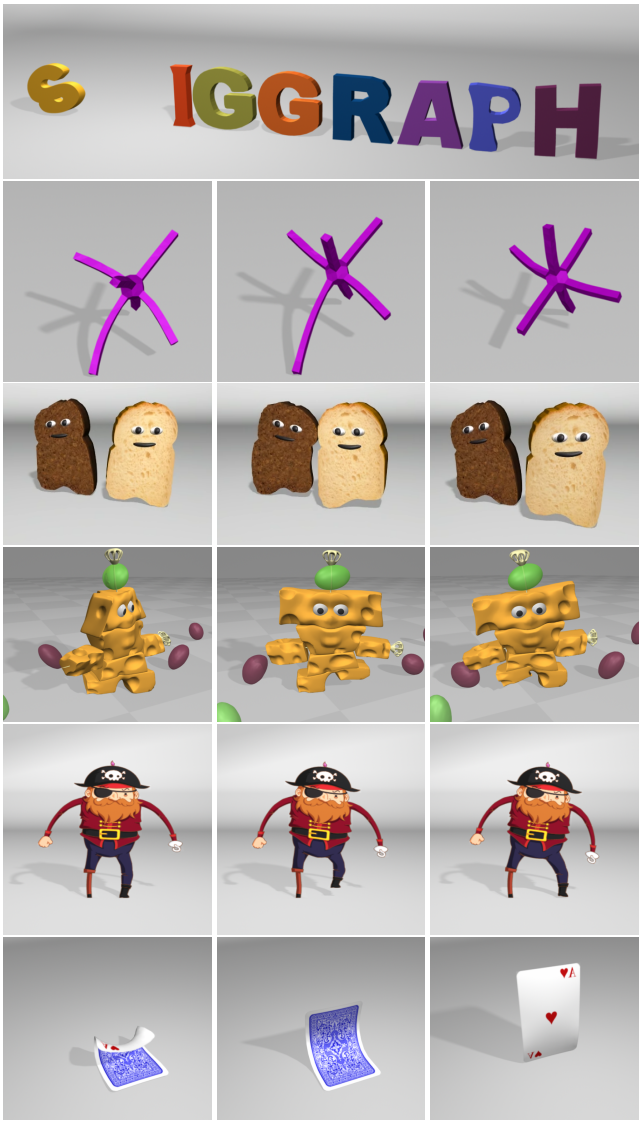

Figure 6: *Deformable objects in action. In row order: Letters moving towards their designated locations.* 3D Cross *uses his extendable legs for fast locomotion. Two* Crusty *characters race each other.* Cheesy *runs away from a bunch of rampaging grapes.* Pirate *walks around in a 2D world.* Ace*'s trick.*

8 Limitations and Future Work

The control framework that we propose automatically changes the rest configuration of deformable objects to allow them to propel themselves through purposeful physical interactions with the environment. We demonstrate the effectiveness of our approach by animating a variety of characters that can crawl, roll, hop and walk.

Our method is not without limitations. The evolution of the state of the dynamical system is currently coupled with the control method. As such, our controller always uses a planning horizon that is equal in length with the time step of the simulation. Characters that are exceedingly soft cannot be well-controlled by our framework, as the effect of changing the rest pose may be too weak over one time step. Adapting a space-time optimization method to work with our problem formulation is one way in which we can increase the planning horizon.

Adapting the rest pose is one way of modulating the internal energy of deformable objects. An alternative is to change the material stiffness parameters. This is a very interesting extension to our work. Currently, for instance, if a character is too soft for its weight, then it would be unable to stand or walk. Automatically adjusting stiffnesses in a non-homogeneous way could address this limitation.

The example-based rest shape adaptation strategy we propose allows users to control the range of internal deformations that the objects are allowed to undergo. This is a key ingredient in controlling the motions of a character, but the motion objectives output by the controllers play an equally important role. The timing of a walk cycle, the motions of the arms or the facial expressions, for instance, can add to the personality conveyed by a character. In future work we plan to further investigate ways in which we can increase the level of artistic control over the resulting motions.

9 Acknowledgements

We would like to thank Alec Jacobson, Fabian Hahn, Joe Schmid, Olga Sorkine and the anonymous reviewers for their useful comments and suggestions. Many thanks to Maurizio Nitti and Alessia Marra for creating the *Cheesy* and *Pirate* characters.

References

BARBIČ, J., AND POPOVIĆ, J. 2008. Real-time control of physically based simulations using gentle forces. In *Proc. of ACM SIGGRAPH Asia '08.*

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

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. TRACKS: Toward Directable Thin Shells. In *Proc. of ACM SIGGRAPH '07.*

BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. PriMo: Coupled prisms for intuitive surface modeling. In *Proc. of Symp. on Geometry Processing (SGP '06).*

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2010. Generalized biped walking control. In *Proc. of ACM SIGGRAPH '10.*

DE LASA, M., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-based locomotion controllers. In *Proc. of ACM SIGGRAPH '10.*

FRÖHLICH, S., AND BOTSCH, M. 2011. Example-driven deformations based on discrete shells. *Comput. Graph. Forum 30*, 8, 2246–2257.

GIRARD, M., AND MACIEJEWSKI, A. 1985. Computational modeling for the computer animation of legged figures. In *Proc. of ACM SIGGRAPH '85.*

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA '03).*

HODGINS, J., WOOTEN, W., BROGAN, D., AND O'BRIEN, J. 1995. Animating human athletics. In *Proc. of ACM SIGGRAPH '95.*

IJIRI, T., TAKAYAMA, K., YOKOTA, H., AND IGARASHI, T. 2009. Procdef: Local-to-global deformation for skeleton-free character animation. In *Proceedings of Pacific Graphics '09.*

IRVING, G., SCHROEDER, C., AND FEDKIW, R. 2007. Volume conserving finite element simulations of deformable models. In *Proc. of ACM SIGGRAPH '07*.

JAIN, S., AND LIU, C. K. 2011. Controlling physics-based characters using soft contacts. In *Proc. of ACM SIGGRAPH Asia '11*.

JEON, H., AND CHOI, M.-H. 2007. Interactive motion control of deformable objects using localized optimal control. In *IEEE International Conference on Robotics and Automation*, 2582 – 2587.

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

KIM, J., AND POLLARD, N. S. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Trans. Graph. 30*.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. In *Proc. of ACM SIGGRAPH '10*.

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

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. In *Proc. of ACM SIGGRAPH '04*.

NOCEDAL, J., AND WRIGHT, S. J. 2000. *Numerical Optimization*. Springer.

O'BRIEN, J., 2011. Thoughts on physically based animation. Keynote talk, Symposium on Computer Animation (SCA).

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proc. of ACM SIGGRAPH '00*.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. In *Proc. of ACM SIGGRAPH '91*.

SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculotendon simulation for hand animation. In *Proc. of ACM SIGGRAPH '08*.

TAN, J., GU, Y., TURK, G., AND LIU, C. K. 2011. Articulated swimming creatures. In *Proc. of ACM SIGGRAPH '11*.

TERAN, J., SIFAKIS, E., BLEMKER, S. S., NG-THOW-HING, V., LAU, C., AND FEDKIW, R. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11*, 317–328.

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

TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: physics, locomotion, perception, behavior. In *Proc. of ACM SIGGRAPH '94*.

TWIGG, C. D., AND JAMES, D. L. 2007. Many-worlds browsing for control of multibody dynamics. In *Proc. of ACM SIGGRAPH '07*.

TWIGG, C. D., AND KAČIĆ-ALESIĆ, Z. 2011. Optimization for sag-free simulations. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA '11)*.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. In *Proc. of ACM SIGGRAPH '10*.

WINKLER, T., DRIESEBERG, J., ALEXA, M., AND HORMANN, K. 2010. Multi-scale geometry interpolation. In *Proc. of Eurographics '10*.

WOJTAN, C., MUCHA, P. J., AND TURK, G. 2006. Keyframe control of complex particle systems using the adjoint method. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA '06)*.

WU, J.-C., AND POPOVIĆ, Z. 2003. Realistic modeling of bird flight animations. In *Proc. of ACM SIGGRAPH '03*.

WU, J., AND POPOVIĆ, Z. 2010. Terrain-adaptive bipedal locomotion control. In *Proc. of ACM SIGGRAPH '10*.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple biped locomotion control. In *Proc. of ACM SIGGRAPH '07*.

A Elastic Models

The elastic potentials used in this work are standard except for a few modifications made to obtain more efficient higher-order derivatives. We use a computer algebra software to automatically generated code for all required derivatives and therefore only list the energy expressions here. In order to compute the energy of a given element, we first compute its energy density using its deformed and undeformed positions $\mathbf{x}$ and $\mathbf{X}$ and then integrate over its initial geometry defined by $\bar{\mathbf{X}}$. While $\mathbf{x}$ and $\mathbf{X}$ change during simulation, $\bar{\mathbf{X}}$ is assumed constant.

For curves in 2D-space, we define the elastic potential as

$$W_{\text{Curve2D}} = \sum_i \frac{1}{2} k_l (l_i^2 - L_i^2)^2 \frac{1}{\bar{L}_i^3} + \sum_a \frac{1}{2} k_b (\theta_a - \Theta_a)^2 \bar{T}_a \ ,$$

where l_i/L_i is the deformed/undeformed length of edge i, θ_a/Θ_a is the deformed/undeformed angle enclosed by two adjacent edges i and j, and $\bar{T}_a = \frac{1}{2}(\bar{L}_i + \bar{L}_j)$. Note that the division by $\bar{L}_i^3$ accounts for the proper scaling of the energy with respect to mesh resolution. We use an analogous formulation for thin shells, replacing θ and Θ with the dihedral angle between two edge-adjacent triangles (see [Grinspun et al. 2003]).

The potential for 2D solids is modeled using Finite Elements and a modified St.Venant-Kirchhoff material similar to [Martin et al. 2011],

$$W_{\text{Solid2D}} = \sum_e \frac{1}{2} k_l ||\mathbf{E}_e||^2 \bar{A}_e + \frac{1}{2} k_a (a_e - A_e)^2 \frac{1}{\bar{A}_e} \ ,$$

with the Green strain $\mathbf{E} = \frac{1}{2}(\mathbf{F}^t\mathbf{F} - \mathbf{I})$ and a/A denoting the area of the deformed/undeformed triangle. The deformation gradient $\mathbf{F}$ is computed as described in Sec. 5.

We also use a straightforward extension of this formulation for 3D volumetric solids but, though tractable, computations become quite intense. For this reason, we constructed a simplified energy that quantifies the distortion of a tetrahedron by measuring the change in edge lengths as

$$W_{\text{Solid3D}} = \sum_e \sum_{j=1}^{6} \frac{1}{12} k_l (l_{ej}^2 - L_{ej}^2)^2 \frac{\bar{V}_e}{\bar{L}_{ej}^4} + \frac{1}{2} k_v (v_e - V_e)^2 \frac{1}{\bar{V}_e} \ .$$