TRACKS: Toward Directable Thin Shells

Miklós Bergou *
Columbia University

Saurabh Mathur *
Columbia University

Max Wardetzky †
Freie Universität Berlin

Eitan Grinspun *
Columbia University

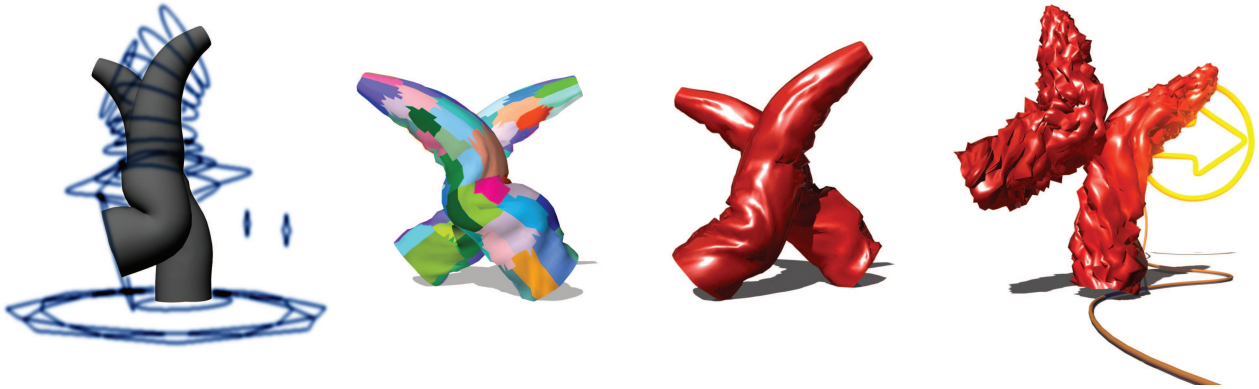


Figure 1: Tracking enables artistic expression and physical simulation to work hand-in-hand, as demonstrated in our animation of a character's unfortunate event. We begin (*left to right*) with the artist's animation, automatically generate a set of Petrov-Galerkin *test functions* (visualized as colored patches), and then solve the constrained Lagrangian mechanics equations to flesh out wrinkles and folds.

Abstract

We combine the often opposing forces of artistic freedom and mathematical determinism to enrich a given animation or simulation of a surface with physically based detail. We present a process called *tracking*, which takes as input a rough animation or simulation and enhances it with physically simulated detail. Building on the foundation of constrained Lagrangian mechanics, we propose *weak-form constraints* for tracking the input motion. This method allows the artist to choose where to add details such as characteristic wrinkles and folds of various thin shell materials and dynamical effects of physical forces. We demonstrate multiple applications ranging from enhancing an artist's animated character to guiding a simulated inanimate object.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: directable animation, tracking, rigging, Galerkin, thin shells

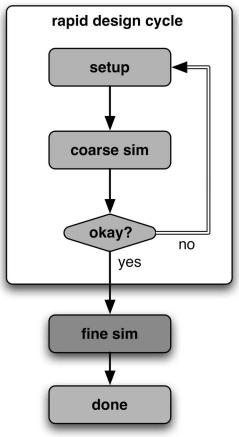
1 Introduction

Simulating thin, flexible materials often means giving up artistic control, yet manually animating their fine folds and wrinkles is an arduous task. How can we provide simultaneous artistic control *and* physical realism for materials like cloth, leather, or metal?

We present a process called *tracking*, which begins with a rough animation already set by the artist and uses physical simulation to add fine-scale details without deviating from the artist's intentions. The artist sets the scale of features to be left intact, and our solver computes the equations of motion at the remaining finer scales.

*e-mail: {miklos|sm2545|eitan}@cs.columbia.edu
†e-mail: wardetzky@mi.fu-berlin.de

Motivating scenarios Consider two scenarios where tracking is important: (a) fleshing out a rough preview of a physical simulation and (b) adding physical detail to an animated character.



Coarse-to-fine design cycle To accelerate the simulation design process, artists use large time steps and coarse geometry to generate rapid previews before committing resources to a full-detail physical simulation (see Fig. 2). When the technical director approves a promising preview, one might consider reusing the parameters of the preview in a full-resolution simulation; unfortunately, an ordinary simulator's output often does *not* resemble the preview. Our tracking solver, on the other hand, *guarantees* a similarity between input and output while adding physically simulated detail at fine scales.

Enriching animation with physics Our work takes one step toward *colocating* animated and physical behavior (see Fig. 1). The animation depicts a puppet-like character whose body consists of a thin, flexible material governed by the laws of physics. In this scenario, there is no distinct spatial or temporal boundary separating art from physics. This must be contrasted with common instances of *disjoint* couplings, *e.g.*: skeleton-driven simulation, simulated fur, or cloth over an animated body (spatially disjoint); and animated keyframes interpolated by physics-based optimization (temporally disjoint). To the best of our knowledge, the spatial and temporal colocation of artistic animation and thin shell physics has not been an explicit goal of prior work in the simulation literature.

1.1 A tracking solution

The scenarios we target in this paper have two main characteristics. On the one hand, we focus on materials governed by the so-called

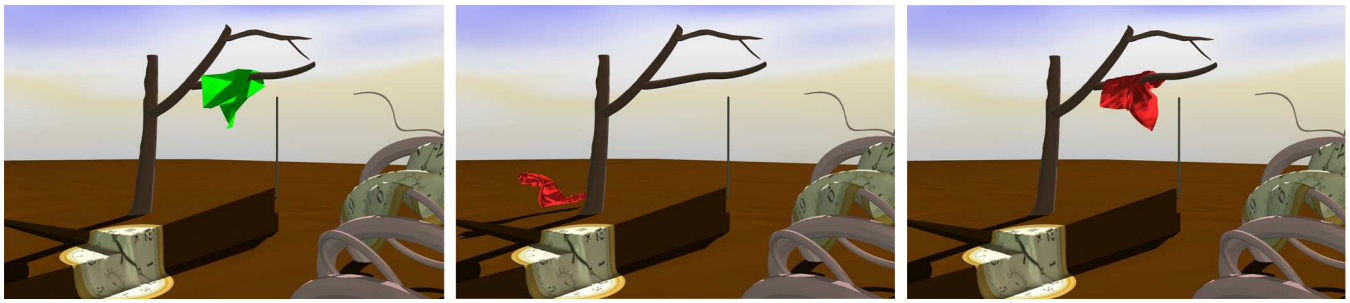


Figure 2: *Rapid simulation design using coarse previews.* Fast simulations with a low-resolution mesh (*left*) enable the technical director to quickly iterate and improve on the setup of a physical simulation. When design is over and final production begins, an attempt to reuse the setup with higher-resolution mesh (*middle*) vividly justifies the usual disclaimer that "past performance is no guarantee of future results." Tracking (*right*) ensures that the output performs as "predicted" by the preview—more precisely, as *prescribed* by the coarse input motion.

thin shell equations: thin shells are flexible surfaces that evolve in time via stretching and bending modes. On the other hand, we assume a given rough input motion—the *guide trajectory*, which our tracking solver enhances with missing material-specific physical detail[1]. The material-dependent physical detail of thin shells expresses itself in characteristic folds, wrinkles, and creases—time-evolving geometry that is exceedingly tedious to model manually. Because these details are straightforward to simulate with mathematical models, thin shell problems present an ideal playground to explore the idea of tracking. Since thin shells wrinkle and fold at multiple spatial and temporal scales, we must address the question of which physical scales to introduce over the given guide trajectory; in general, the answer to this question may vary over space and time. We provide the artist with several intuitive controls to describe the *coarseness* of the input trajectory and to set up the space of permissible physical *details*.

Method-in-brief We propose a simple framework for tracking solvers, or *TRACKS*, which enhances a given guide trajectory with physically simulated detail. The framework consists of:

1. establishing a correspondence between *guide* (input) and *tracked* (output) shapes (see §4.1),
2. creating weak-form *averaged constraints* using Petrov-Galerkin test functions (see §3.1 and §4.2), and
3. solving the *Constrained Lagrangian Mechanics* equations to add physical detail while enforcing the matching constraints (see §3.2 and §4.3).

The key feature that distinguishes our method from previous work lies in our use of the weak form formulation to create constraints that force the guide and tracked trajectories to match in an overall or averaged sense. To facilitate the creation of the required correspondence, we also provide a novel extension to Lloyd's algorithm that automates the process. To the best of our knowledge, our solver is the first to flesh out a given surface animation with simulated dynamic detail, in a single pass, with artistic control over the scale of introduced details. However, our work builds on a large body of preceding contributions, which we briefly survey below.

2 Related work

The roots of our work trace back to seminal papers by Terzopoulos *et al.* [1987; 1988] on physical simulation of deformable models, Platt and Barr [1988] on constrained Lagrangian mechanics, and Witkin and Kass [1988] on spacetime constraints. In the subsequent two decades, numerous works developed methods for guiding the course of a physical simulation. The directing of fluid simulations was considered by McNamara [2004], Fattal [2004], Rasmussen [2004], Shi [2005], Angelidis [2006], Thürey [2006], and their respective co-workers. Other research focused on controlling the movement of elastic solids; see, *e.g.*, [Smith et al. 2001; Capell et al. 2002; Capell et al. 2005; Kondo et al. 2005; Sifakis et al. 2005].

In contrast, work on directing plate and shell dynamics is comparatively scarce. Cutler *et al.* [2005] and Bridson *et al.* [2003] presented techniques for prescribing wrinkles on worn garments. Recently, Wojtan *et al.* [2006] applied the adjoint method to a cloth particle system; their work builds on ideas of control using spacetime constraints:

Control using multiple-pass and single-pass methods The *spacetime constraints* paradigm [Witkin and Kass 1988] asks for a trajectory that minimizes the work required to satisfy user-provided *keyframes*. Methods built on this paradigm can produce convincing motion based on very sparse data (a few constraints). The requisite solver—typically a form of multiple-shooting [Witkin and Kass 1988; Popović et al. 2003] or gradient-based optimization [McNamara et al. 2004]—can be computationally expensive: each iteration costs on the order of one simulation, and the number of iterations grows with the size of the *spacetime window*. Multiple-pass methods such as these may be intractable as the complexity of the physical system increases. To alleviate these problems, Cohen [1992] and later Treuille *et al.* [2003] applied windowing and refinement strategies.

Seeking to avoid these limitations, various authors proposed single-pass approaches based on *guide particles* [Rasmussen et al. 2004] and *prescribed force fields* [Sifakis et al. 2005; Capell et al. 2005]. Guide-particle methods have focused on fluids, and we are not aware of an application to thin plates and shells that is free of tugging or other artifacts (see Fig. 4). In §3.2, we interpret our method in the framework of a force field approach by regarding the constrained Lagrangian as a means to automatically determine fictitious guide forces. Our work fits the single-pass paradigm: a tracking solver has runtime cost on the order of an ordinary simulation.

Tracking Popović *et al.* [2000; 2003] and Kondo *et al.* [2005] presented systems for rigid and deformable bodies in which a user sketches a trajectory and sets up key poses, and a physics solver

[1] Here we use the term *trajectory* as the time-evolving shape and position of the surface—as opposed to the path of its center of mass.

produces a conforming animation. Motivated by this work, we introduce ideas from multiresolution, enabling our thin shell solver to add wrinkles and folds without disturbing the coarse motion. Employing multiresolution in a physical simulation was previously considered in [Grinspun et al. 2002], which adaptively refined basis functions to represent geometric detail. In contrast, our work uses test functions for defining spatially-averaged constraints, effectively fixing the *space* of permissible details. While both of these works use multiresolution in a physical context, several related works do so in a geometric context:

Enriching surface animations Recently, Kircher and Garland [2006] presented a multiresolution signal processing framework for editing surface animations, which included superimposing detail on a moving surface. Hadap *et al.* [1999] and Ma *et al.* [2006] added wrinkles to worn garments, and Loviscach [2006] presented a GPU-based method. These works adopted a *geometric* paradigm suited to a wide array of interactive tasks. Galoppo *et al.* [2006] introduced a physically based method for simulating high-resolution surface contact and collisions. However, we are not aware of prior work that uses physical simulation to add dynamic, material-dependent wrinkles and folds to an existing animation.

Reuse and data-driven methods Our solver enables an artist to reuse a coarse animation in achieving many different possible appearances for the output. Park and Hodgins [2006] acquired and then reproduced surface wrinkles. Cordier and Magnenat-Thalmann [2005] developed an interactive cloth simulator that reuses wrinkles learned from offline simulations. Both of these methods *reuse* geometric information; they belong to a broader category of approaches that encode and reproduce surface geometry relative to a subject's pose (see [Singh and Kokkevis 2000; Sumner and Popović 2004] and references therein). Similarly, reuse of motion sequences was considered by, *e.g.*, [Zordan and Hodgins 2002; Gleicher et al. 2003].

3 Control via Petrov-Galerkin constraints

In this section, we explain our central idea: matching the tracked output trajectory to the user-provided input trajectory by using Petrov-Galerkin constraints. We start with a simple 1D example, showing the difference between pointwise (interpolating) and weak (averaged) constraints. In §3.1, we make precise the formulation of weak constraints, and in §3.2, we show how to enforce these constraints using the framework of Constrained Lagrangian Mechanics.

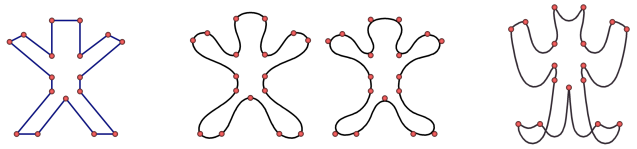


Figure 3: *Interpolating vs. averaging.* Starting with a guide shape (*left*), interpolating and averaging approaches (*center, respectively*) both produce pleasing results in the absence of dynamics. Adding a sharp acceleration, however, results in tugging artifacts for the interpolating approach (*right*), while the results obtained via averaging are unchanged.

Motivation We begin with the experiment shown in Fig. 3, in which we deform a rubber band (*i.e.*, an elastic curve governed by bending and stretching forces) to "look like" the given guide shape. The *interpolating* approach forces the rubber band to pass through certain anchor points (shown in red) on the guide shape. On the other hand, the *averaging* approach asks for certain mean quantities (such as the centers of mass of corresponding segments between anchors) to be equal, which leads to an approximating curve not passing through the anchor points. As shown in Fig. 3, the interpolating approach can lead to undesirable tugging artifacts. The value of averaging in avoiding these artifacts is widely recognized in the geometric modeling community [Zorin and Schröder 1998].

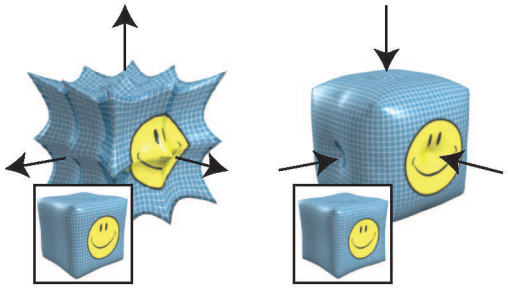


Figure 4: *Artifacts of pointwise constraints.* As the guide shape rapidly expands and contracts, pointwise constraints produce high-frequency tugging artifacts, whereas our averaged constraint method (*inset*) tracks smoothly.

In Fig. 4, we show a similar experiment for a 2D surface, with much the same results as in the 1D setting. This example uses more complicated averaged constraints than equating centers of mass. To better understand this, we make precise the notion of an averaged constraint.

3.1 Constraint formulation

Consider the guide and tracked configurations evolving over time in 3-space, given respectively by

$$\bar{\mathbf{q}} : \Omega \times \mathbb{R} \to \mathbb{R}^3 \quad \text{and} \quad \mathbf{q} : \Omega \times \mathbb{R} \to \mathbb{R}^3 .$$

Here Ω is the so-called *reference* or *material domain* of the elastic body, $\bar{\mathbf{q}}$ is the deformation mapping of the *guide* configuration, and $\mathbf{q}$ is the deformation mapping of the *tracked* configuration from the material domain into 3-space [Malvern 1969]. We assume that the guide configuration, $\bar{\mathbf{q}}(\cdot,t)$, is given, whereas we view the tracked configuration, $\mathbf{q}(\cdot,t)$, as the temporally-evolving configuration of a physical system. For the remainder of this section, we focus on a fixed instant in time and omit the time argument.

We subject the tracked physical system to *holonomic constraints* by restricting its motion to the space of permissible configurations, $\{\mathbf{q} \in \mathbf{Q} \,|\, \mathbf{g}(\mathbf{q}) = 0\}$, for some configuration space $\mathbf{Q}$ and objective function $\mathbf{g}$ [Lanczos 1986]. In other words, we only consider solutions satisfying the constraint equation $\mathbf{g}(\mathbf{q}) = 0$ at all instants in time. Below, we define $\mathbf{g}(\mathbf{q})$, and in §3.2 we explain how to enforce $\mathbf{g}(\mathbf{q}) = 0$ during simulation.

In choosing $\mathbf{g}$ we encapsulate the requirement that the tracked and guide shapes match at certain coarse spatial scales. We define a *set* of vector-valued constraints $\{\mathbf{g}^1, \mathbf{g}^2, \dots\}$ by

$$\mathbf{g}^i(\mathbf{q}) = \int_\Omega \bar{\mathbf{q}}(x)\phi_i(x)\mathrm{d}x - \int_\Omega \mathbf{q}(x)\phi_i(x)\mathrm{d}x \,. \qquad (1)$$

This is a set of Petrov-Galerkin equalities, formed by choosing a specific set of *test functions* $\{\phi_1, \phi_2, \dots, \phi_K : \Omega \to \mathbb{R}\}$ [Strang and Fix 1973]. For every ϕ_i, we constrain separately the three coordinate functions of the embeddings $\bar{\mathbf{q}}$ and $\mathbf{q}$. Simply stated, each equation requires a weighted average of positions on the guide surface to match a weighted average of positions on the tracked surface, using $\phi_i(x)$ as the weighting function.

We call attention to some of the properties of these averaged constraints: (i) they are *linear* in the unknown positions, $\mathbf{q}$; (ii) they

depend explicitly on time since the guide shape moves over time; and (iii) the space of permissible configurations is closed under simultaneous rigid transformation of the tracked and guide shapes.

From a *Petrov-Galerkin* perspective, (1) requires a *weak* identification of tracked and guide coordinate functions under the finite-dimensional test space spanned by $\{\phi_i\}$. However, whereas classically the space of test functions is chosen based on differentiability requirements and then held fixed throughout the computation [Strang and Fix 1973], we consider the support and profile of the test functions as variable in space and time. Indeed, the choice of test functions is a problem in *filter design*, if we consider that (1) requires the low-pass filtered version of the guide geometry to match the low-pass filtered version of the tracked geometry. The filter size and shape are linked to the support of the test functions: the proximity of the tracked to the guide shape increases with the locality of test function support (see Figs. 5 and 6).

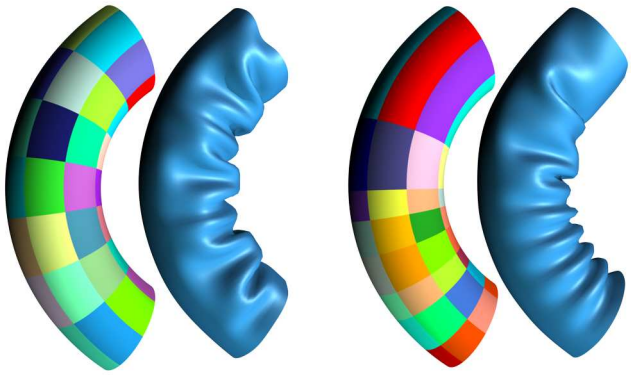


Figure 5: *Spatially varying test functions.* Buckling of the tracked surface is directly linked to the distribution of test functions over the mesh. A homogeneous distribution (*left*) results in uniform wrinkling of the surface, whereas a heterogeneous distribution (*right*) causes larger folds to form in coarser regions and smaller wrinkles to form in finer-support regions.

Thus far, we have assumed the existence of a common reference surface, Ω. In practice, explicit access to the material domain may not be possible, *e.g.*, because the software is not finite-element based, or the user asks for tracked and guide geometry of differing genus. Therefore, we now relax this assumption.

Consider $\bar{\mathbf{q}}$ and $\mathbf{q}$ redefined over two *distinct* reference configurations, $\bar{\Omega}$ and Ω. In practice, we use the undeformed shape as the reference configuration. Recall that in (1) the test basis induced a set of weak equalities evaluated over a common domain. Having discarded the common domain, we now need a set of mutually *corresponding test functions*, $\{\bar{\phi}_i\}$ over $\bar{\Omega}$, and $\{\phi_i\}$ over Ω (for three ways to establish this correspondence, see §4.1). Then the resulting constraint equations are given by

$$\mathbf{g}^i(\mathbf{q}) = \int_{\bar{\Omega}} \bar{\mathbf{q}}(\bar{x})\bar{\phi}_i(\bar{x})\mathrm{d}\bar{x} - \int_{\Omega} \mathbf{q}(x)\phi_i(x)\mathrm{d}x \,, \tag{2}$$

where each test function must satisfy the normalization condition

$$\int_{\bar{\Omega}} \bar{\phi}_i(\bar{x})\mathrm{d}\bar{x} = \int_{\Omega} \phi_i(x)\mathrm{d}x \,. \tag{3}$$

This condition is always satisfiable by rescaling each ϕ_i. When $\bar{\Omega}$ and Ω have different area measures, $\mathrm{d}\bar{x}$ and $\mathrm{d}x$, then normalization ensures that the space of permissible configurations remains closed under simultaneous rigid transformation of the tracked and guide shapes. We now turn our attention to enforcing the constraints given by (2) and (3) in a physical simulation.

3.2 Constrained Lagrangian Mechanics

TRACKS takes as input a user-provided guide to the object's motion and produces as output a physically detailed refinement of that motion. At each instant in time, the solver evolves the tracked shape under its governing equations, which include physical forces as well as averaged constraints. To enforce these constraints, we chose the method of *Lagrange multipliers*, which is particularly well suited for a Constrained Lagrangian Mechanics (CLM) approach.

In constrained mechanics, a physical system must satisfy both the laws of motion and the constraint of staying on a certain geometric configuration [Platt and Barr 1988]. By adopting the Lagrangian viewpoint of mechanics, our method chooses the most physical trajectory of all possible trajectories that satisfy the constraint equations [Marsden and Ratiu 1994]. This framework leads to the *constrained Euler-Lagrange equations* [Lanczos 1986]

$$\begin{aligned} M\ddot{\mathbf{q}} - \mathbf{F}(\mathbf{q}) + \lambda^T \frac{\partial \mathbf{g}}{\partial \mathbf{q}} &= 0 \,, \\ \mathbf{g}(\mathbf{q},t) &= 0 \,. \end{aligned} \tag{4}$$

Here, $\ddot{\mathbf{q}}(t)$ is acceleration, $\mathbf{F}$ represents the forces in the system (both those arising from the elastic stretching and bending energy of the surface and any external forces), $\mathbf{g}$ is the objective function (2), and λ is the Lagrange multiplier. Our treatment incorporates m vector-valued constraints, so that $\mathbf{g} \in \mathbb{R}^{3m}$ and $\lambda \in \mathbb{R}^{3m}$.

At each instant in time, (4) defines two sets of equations in the two sets of variables $\mathbf{q}(t)$ and $\lambda(t)$. The first equation represents the physical evolution of the system according to Newton's law, incorporating fictitious constraint-maintaining forces $\lambda^T \frac{\partial \mathbf{g}}{\partial \mathbf{q}}$, whereas the second equation is used to determine those forces such that the constraint is exactly maintained.

We view the method of Lagrange multipliers as a way to automatically choose, at any instant in time, *constraint forces* that are just strong enough to maintain the constraint, but not stronger than necessary. The Lagrange multipliers themselves control the magnitude of the force, and the constraint gradient determines the direction of the force.

Alternative constraint enforcement methods Many equally viable alternatives can be used to enforce (2) during simulation. Different from our approach, the constraints, $\mathbf{g} = 0$, may be *maintained differentially* by requiring that $\dot{\mathbf{g}} = 0$ (see, *e.g.*, [Witkin and Welch 1990]), effectively constraining the allowable accelerations. This approach gives rise to a linear system for determining the Lagrange multipliers, λ, even if the constraints themselves are non-linear. In practice, an additional constraint-correcting force is used to prohibit undesirable drifting of the solution away from the constraint manifold and to bring the initial state into an allowable configuration [Witkin and Baraff 2001]. Since (2) defines *linear* constraints, we find it straightforward to work with $\mathbf{g} = 0$ directly, which entirely eliminates the problem of drifting.

Alternatively to Lagrange multipliers, we could have used *penalty springs* to "pull" the system onto the constraint manifold. The problem with this approach is that it requires a predetermined stiffness coefficient for each constraint, which can be difficult to choose. Likewise, critically damping a penalty force can be difficult. For these reasons, the penalty method may suffer from unnecessary numerical stiffness, instability, or unsatisfied constraints [Platt and Barr 1988; Witkin and Baraff 2001]. Using Lagrange multipliers entirely circumvents these problems.

4 Building a tracking solver

In this section, we begin by describing how to design and implement the weak-form constraints when using meshes to represent the guide and tracked surfaces, then show a numerical integration scheme for the equations of motion that maintain those constraints, and conclude by pointing the reader to thin shell simulation models that can be used within this framework.

4.1 Designing test functions

Recall from (2) that in order to create the tracking constraints, we need to define a matching between the guide and tracked configurations by establishing *corresponding* test functions over both surfaces. We provide one manual and two automatic methods to design such a correspondence for meshes.

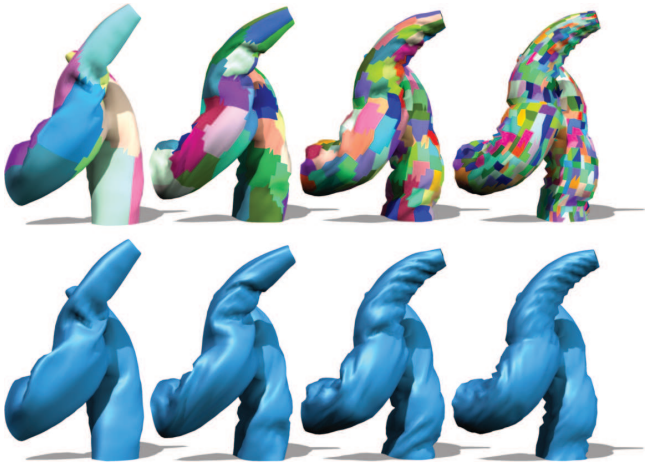


Figure 6: *Constraint resolution.* Increasing the number of partitions (*left-to-right:* 16, 64, 256, 1024) narrows the support of the test functions, forcing closer tracking and finer wrinkles.

On-mesh painting In the most hands-on approach, the user paints pairs of corresponding regions on the undeformed (reference) guide and tracked meshes. To each painted region on the guide mesh, we associate a test function ϕ_i defined to have value 1 for all vertices inside that region and value 0 otherwise. To determine the constant value of ϕ_i within the corresponding region on the tracked mesh, we use the normalization condition given by (3).

Mesh subdivision For some examples, (see Figs. 4, 8, and 9) we create the undeformed tracked mesh by linearly subdividing the undeformed guide mesh; such subdivision automatically induces a hierarchical basis [Zorin and Schröder 1998]. We use the coarsest level of the resulting hierarchical basis as the test functions on both the guide and tracked meshes, so $\phi_i = \bar{\phi}_i$. These test functions correspond to the Lagrange basis functions on the guide mesh: each function ϕ_i is linear and defined to be 1 at vertex i on the guide mesh and 0 at all others.

Variational clustering Motivated by Variational Shape Approximation (VSA) [Cohen-Steiner et al. 2004], we propose an animation-aware extension to Lloyd's algorithm to partition the guide mesh into r disjoint regions (see Fig. 6) and assign a test function to each region. To initialize the algorithm, we randomly choose r triangles as *proxies* $\{P_1, \ldots, P_r\}$. The algorithm then proceeds in two steps: (i) growing regions and (ii) updating the proxies.

Step i: We grow regions around the proxies using the algorithm described in §3.3 of [Cohen-Steiner et al. 2004], which assigns each mesh triangle to the *closest* proxy while ensuring connectedness of regions. To compute the distortion distance between a guide mesh triangle, T_j, and a proxy, P_k, we use the error metric $E(T_j, P_k) =$

$$\sup_{t \in [t_0, t_1]} A_j(t) \left(\frac{\mu}{A(t)} \|\mathbf{x}_j(t) - \mathbf{x}_k(t)\|^2 + (1-\mu) \|\mathbf{n}_j(t) - \mathbf{n}_k(t)\|^2 \right) ,$$

where $\mathbf{x}_j(t)$ and $\mathbf{x}_k(t)$ are centroids of T_j and P_k respectively, $\mathbf{n}_j(t)$ and $\mathbf{n}_k(t)$ are unit normals associated to T_j and P_k respectively, $A_j(t)$ is the area of T_j, $A(t)$ is the total surface area of the guide mesh, and all the aforementioned variables are functions of time. The user-prescribed coefficient $\mu \in [0,1]$ controls the bias of the regions toward compactness ($\mu \to 1$) or flatness ($\mu \to 0$).

Step ii: Next, we recompute the proxy, P_k, for each region as an area-weighted average of its constituent triangles' position and normal; since geometry is a function of time, so are the proxies.

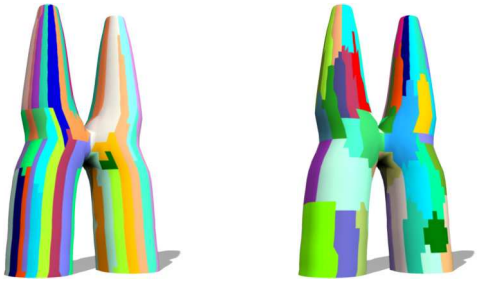


Figure 7: *Animation-aware clustering.* Results shown using original *(left)* and animation-aware *(right)* VSA clustering algorithm.

These two steps are repeated until convergence, resulting in a clustering of the guide mesh into regions (see Fig. 7). We use this partitioning algorithm only when the undeformed tracked and guide meshes are identical, so the regions on the tracked mesh are the same as those on the guide. Once these regions are created, we again assign a test function ϕ_i to each region on the mesh such that $\phi_i = 1$ for vertices within the region and $\phi_i = 0$ otherwise (since the two meshes are identical, we have $\phi_i = \bar{\phi}_i$).

Our error metric differs from that proposed in VSA due to two considerations: First, whereas VSA's focus on *remeshing* prefers large, flat regions, our focus on *tracking* prefers evenly-sized, flat regions; consequently, we use the Euclidean metric in place of VSA's distance-to-plane metric. Second, whereas VSA partitions a static mesh, we seek a fixed partition that is reasonable at any instant in time, $t \in [t_0, t_1]$, for the given animation of the guide mesh. Therefore, we include a supremum over time. In practice, we approximate the supremum by considering a maximum over a fixed set of randomly chosen frames. Notice that our animation-aware algorithm tends to align seams between regions to boundaries between rigidly deforming patches, *e.g.*, to the character's joints (see Fig. 7).

For the 3038-vertex Xavier (see Fig. 1), our animation-aware partitioning required 32 iterations (less than five minutes) to generate a 200-region partition. The choice of cluster-count effectively controls the cut-off point between the scale of the details added through simulation and those kept intact from the input guide.

4.2 Implementing constraints

In our implementation, we express the test functions on the guide mesh as linear combinations of the Lagrange basis functions, $\{\bar{\psi}_j\}$:

$$\bar{\phi}_i(\bar{x}) = \sum_{j=1}^{\bar{n}} \bar{S}_{ij}\,\bar{\psi}_j(\bar{x}) \;, \tag{5}$$

where $\bar{n}$ represents the number of vertices in the guide mesh. To obtain the entries of the matrix $\bar{S}$, we use one of the methods from §4.1 to evaluate ϕ_i at mesh vertices and set $\bar{S}_{ij} = \bar{\phi}_i(\bar{x}_j)$, where $\bar{x}_j$ is the material coordinate of vertex j of the guide mesh. We write the test functions, $\{\phi_i\}$, on the tracked mesh in an analogous manner to (5).

We can now express the $3m$ constraints from (2) and the associated gradient at time step k as

$$\mathbf{g}_k(\mathbf{q}_k) = \underbrace{\bar{S}\bar{\mathscr{M}}}_{\bar{C}}\bar{\mathbf{q}}_k - \underbrace{S\mathscr{M}}_{C}\mathbf{q}_k \quad \text{and} \tag{6}$$

$$\nabla\mathbf{g}_k(\mathbf{q}_k) = -C \;, \tag{7}$$

where we use the abbreviation $\mathbf{g}_k(\mathbf{q}_k) = \mathbf{g}(\mathbf{q}_k, t_k)$, and the entries of the Petrov-Galerkin mass matrix, $\mathscr{M}$, are given by

$$\mathscr{M}_{jj} = \frac{A_j}{6} \qquad \text{and} \qquad \mathscr{M}_{jl} = \frac{A_{jl}}{12} \;.$$

Here, A_j denotes the total area of the triangles incident to vertex j, and A_{jl} denotes the total area of the triangles incident to edge jl (and analogously for $\bar{\mathscr{M}}$).

The normalization condition from (3) can be satisfied by scaling the entries in each row of C at the beginning of the simulation:

$$C_i \leftarrow C_i \frac{\sum_{j=1}^{\bar{n}} \bar{C}_{ij}}{\sum_{l=1}^{n} C_{il}} \;. \tag{8}$$

Since the matrices C and $\bar{C}$ depend only on the undeformed meshes and the test functions, they can be precomputed for efficiency.

4.3 Numerical integration

We describe our implementation of the constrained implicit Euler method (for alternatives consult [Hauth et al. 2003; Boxerman and Ascher 2004; Hairer et al. 2006]):

$$\begin{aligned} M\frac{\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k}{h} - \mathbf{F}(\mathbf{q}_{k+1}) + \lambda_k^T \nabla\mathbf{g}_k(\mathbf{q}_k) &= 0 \;, \\ \frac{\mathbf{q}_{k+1} - \mathbf{q}_k}{h} - \dot{\mathbf{q}}_{k+1} &= 0 \;, \\ \mathbf{g}_{k+1}(\mathbf{q}_{k+1}) &= 0 \;. \end{aligned}$$

Here $(\mathbf{q}_1, \mathbf{q}_2, \ldots)$ denotes the time-discrete trajectory of the fine mesh, $(\dot{\mathbf{q}}_1, \dot{\mathbf{q}}_2, \ldots)$ denotes the velocities along the trajectory, M is the physical mass matrix, and $\mathbf{F}$ represents the forces present in the system (see §4.4). The above system can be rearranged as an implicit equation in the unknown variables $(\dot{\mathbf{q}}_{k+1}, \lambda_k)$, with the update rule for $\mathbf{q}_{k+1}$ given in the second line. We compute a single iteration of Newton's method per time step, as in [Baraff and Witkin 1998]. Taking $(\dot{\mathbf{q}}_k, 0)$ as the initial guess for $(\dot{\mathbf{q}}_{k+1}, \lambda_k)$ yields:

$$\begin{bmatrix} M - h^2 J_F(\mathbf{q}_k + h\dot{\mathbf{q}}_k) & h\,[\nabla g_k(\mathbf{q}_k)]^T \\ h\nabla g_k(\mathbf{q}_k) & 0 \end{bmatrix} \begin{bmatrix} \Delta\dot{\mathbf{q}} \\ \lambda_k \end{bmatrix} = \begin{bmatrix} h\mathbf{F}(\mathbf{q}_k + h\dot{\mathbf{q}}_k) \\ -g_{k+1}(\mathbf{q}_k + h\dot{\mathbf{q}}_k) \end{bmatrix},$$

where $J_F(\mathbf{q}_k + h\dot{\mathbf{q}}_k)$ is the Jacobian of forces evaluated at $\mathbf{q}_k + h\dot{\mathbf{q}}_k$, and the constraint and constraint gradient are evaluated using (6) and (7), respectively. We solve this system for $(\Delta\dot{\mathbf{q}}, \lambda_k)$ using the PARDISO [Schenk and Gärtner 2006] sparse linear solver, then set

$$\dot{\mathbf{q}}_{k+1} = \dot{\mathbf{q}}_k + \Delta\dot{\mathbf{q}} \qquad \text{and} \qquad \mathbf{q}_{k+1} = \mathbf{q}_k + h\dot{\mathbf{q}}_{k+1} \;.$$

4.4 Implementing thin shell simulations

Our approach is not tied to a particular way of discretizing shells or cloth and easily incorporates into an existing solver with its constituent implementation of bending and stretching forces. We chose to work with triangle meshes and employed the model of *Discrete Shells* [Grinspun et al. 2003] for bending potential energy and *Constant Strain Triangle* [Zienkiewicz and Taylor 2000] for in-plane stretching potential energy. As these models (and several variants surveyed in [Ng and Grimsdale 1996; Thomaszewski and Wacker 2006]) are by now well known and widely employed by the graphics community, we refer the reader to the above references for details.

The true power of tracking is most evident when the underlying (non-tracking) simulator is able to capture the look and feel of many different materials. In this light, we briefly describe a simple plasticity model that broadened our simulator's expressive power.

Simple model for plasticity Plasticity has been widely studied within computer graphics: see [Terzopoulos and Fleischer 1988; O'Brien et al. 2002; Irving et al. 2004; Gingold et al. 2004; Wicke et al. 2005] and references therein (for an overview of plasticity, see [Zienkiewicz and Taylor 2000]). We briefly outline a model for plasticity developed primarily with consideration for (a) simplicity of implementation and (b) proper scaling (material response should not depend on mesh- or time-resolution).

We start with the bending model used in [Baraff and Witkin 1998; Bridson et al. 2003; Grinspun et al. 2003] and add two additional parameters: (i) maximal curvature deviation, $\Delta\kappa_{max} \in [0, \infty]$, which dictates the maximal strain for which response is purely elastic and (ii) hardening factor, $\eta \in \mathbb{R}$, which governs the hardening ($\eta > 0$) or softening ($\eta < 0$) of creases due to plastic work. For each mesh edge, we store the undeformed configuration's bend angle, $\bar{\theta}_i$, as well as a per-edge elastic bending stiffness, k_i. At the start of each time step, we test for a *plastic update* on each edge (the subscript i is implied). We compute the *curvature deviation*,

$$\Delta\kappa = \kappa - \bar{\kappa} = \frac{\bar{e}}{2\bar{A}}(\theta - \bar{\theta}) \;,$$

where $\bar{A}$ is the combined area of the undeformed triangles incident to edge $\bar{e}$, and θ (resp. $\bar{\theta}$) is the angle formed between incident deformed (resp. undeformed) triangle normals. Unlike the change in bend angle $(\theta - \bar{\theta})$, the curvature deviation accounts for spatial scaling (mesh sampling density) [Grinspun 2006]. If $|\Delta\kappa| > \Delta\kappa_{max}$, we perform a *plastic strain update* with *plastic hardening*:

$$\begin{aligned} \bar{\theta} &\leftarrow \bar{\theta} + \operatorname{sign}(\Delta\kappa)\frac{2\bar{A}}{\bar{e}}\left(|\Delta\kappa| - \Delta\kappa_{max}\right) \;, \\ k &\leftarrow k\exp\left(\eta\left(|\Delta\kappa| - \Delta\kappa_{max}\right)\right) \;. \end{aligned}$$

We note that the exponential provides proper temporal scaling.

External forces and collisions While we focus mainly on material properties, our method accommodates wind and other external forces. As it is single-pass and involves only forward-dynamics, any existing collision-response code should be compatible with our approach; in contrast, space-time optimization has difficulties under complex collisions [Wojtan et al. 2006]. Our experiments use the robust treatment of collisions and self-collisions described in [Bridson et al. 2002].

No mercy.

5 Applications

In the following application examples, we demonstrate *TRACKS* by directing discrete shell simulations. The computation times for the tracking solver are very practical, consuming for the most complex examples comfortably below four hours on a 2GHz processor. Indeed, we find that, consistent to our goal, the bulk of our time and effort is allocated to the artistic process. We consider scenarios spanning four axes:

- To **create the input trajectory**, we use a rapid-preview physical simulation in §5.1, procedurally generate motion in §5.2, build on a motion-capture (mocap) sequence in §5.3, and ask an artist to animate expressive character gestures in §5.4.
- In regards to **input mesh resolutions**, we test a coarsest possible input in §5.2, a more refined input in §5.1 and §5.3, and a fine input in §5.4.
- In setting up **input→output combinatorics**: the output corresponds to a combinatorially-subdivided input in §5.1, §5.2, and §5.3, whereas input and output are combinatorially identical in §5.4.
- We consider three techniques for **creating test functions**: in §5.2–§5.3 we use the Lagrange basis of the guide mesh, prolongated onto the fine tracked mesh via linear subdivision; in §5.4, we use a novel extension to Lloyd's algorithm to automatically construct a piecewise constant basis; but first, in §5.1, we put aside the automatic methods, and manually paint a pair of test functions.

5.1 Simulation design with fast, coarse previews

Fleshing out a coarsely-run simulation is a prototypical use of our solver. As shown in the accompanying video and Fig. 2, the director intends for the flying cloth to get stuck on a branch. We search for initial conditions and material coefficients using a *rapid preview* design cycle: working with a coarse mesh (35 vertices), after twenty design iterations, or about 100min working on a 2GHz notebook, we establish our desired parameters (see Fig. 2-*left*). If instead, the full-resolution mesh (1625 vertices) were used during design, the consequent simulation cost (30+ hours) would require a batch process, instead of a rapid and thus more intuitive design cycle.

Unfortunately, the preview's parameters do not translate well to the high-resolution simulation as shown in Fig. 2-*middle*. Indeed, despite great care in designing a system of meshing-independent simulation parameters (*i.e.*, a system that incorporates well-reasoned scaling factors), the bifurcations and complex configuration landscape of quasi-inextensible surfaces that buckle and fold make it very unlikely that a coarse and fine simulation will agree.

Using our tracking method ensures that the final, expensive, high-resolution computation produces the expected results on the first run. We use four hand-painted test functions on the flag because our goal is to guarantee a global trajectory for the cloth, not to control its fine shape. The fine mesh is a combinatorially-subdivided version of the input one.

5.2 The simple box

A more challenging scenario is one in which the coarse motion is not physically based. Our simplest such experiment begins with a coarse box modeled with eight vertices (see Fig. 8), which is animated with procedurally-generated rigid motion of the top face while the bottom face is fixed to the ground. We produce the corresponding fine mesh by linearly subdividing four times (triangle quadrisection and a linear geometric stencil), and we choose as our (piecewise linear) test functions the coarsest subdivision basis.

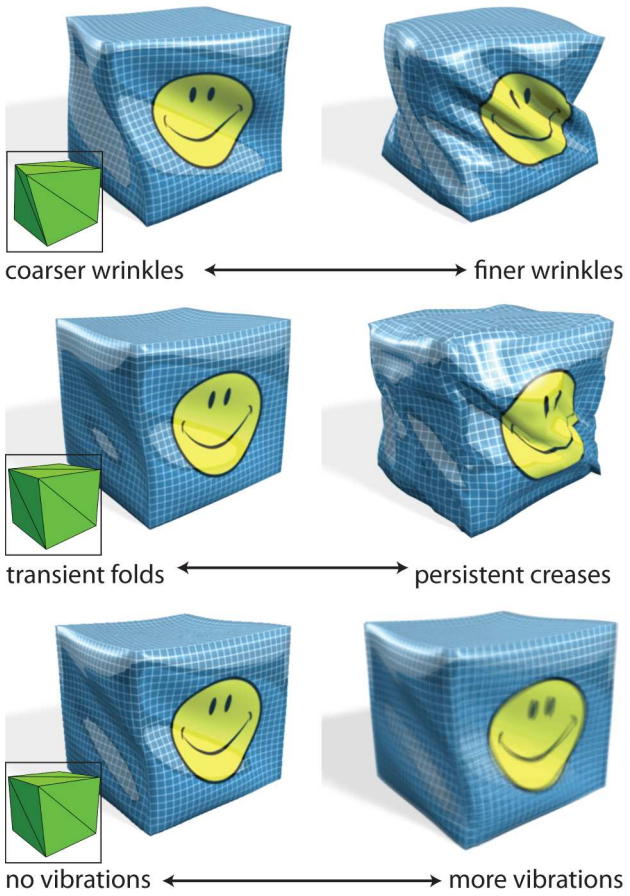


Figure 8: *Material axes.* Traditionally, after an animation is completed, the lighting and rendering crew adjust and compare various rendering alternatives. With *TRACKS*, an artist enjoys the *post facto* freedom to also adjust *material and dynamical* properties: (*top*) bending and stretching stiffness control the shape of wrinkles during a twist; (*middle*) plastic yield and hardening control whether the material rebounds or the wrinkles persist; (*bottom*) damping and mass density control oscillations such as vibrations and wobbliness.

The viewer witnesses an inanimate object come to life, as the fine box simultaneously achieves two objectives: (i) it clearly tracks the non-physical coarse motion while (ii) exhibiting dynamical material properties. By *dynamical*, we mean that similar effects could *not* be achieved by post-processing each frame of the coarse animation in isolation. For example, at the end of the sequence, the coarse mesh suddenly stops moving, yet the tracked mesh continues to move under inertia.

5.3 Motion-captured backflip

By combining multiple constitutive laws and dynamical controls, *TRACKS* enables an artist to apply a broad range of material behaviors to a predefined coarse motion. Since *TRACKS* is a framework incorporated over existing simulation code, it inherits the expressive power and material realism existent in the underlying simulation software.

In this example, we begin with mocap data of a breakdancer's backflip. As depicted in Fig. 9 and the accompanying video, we apply

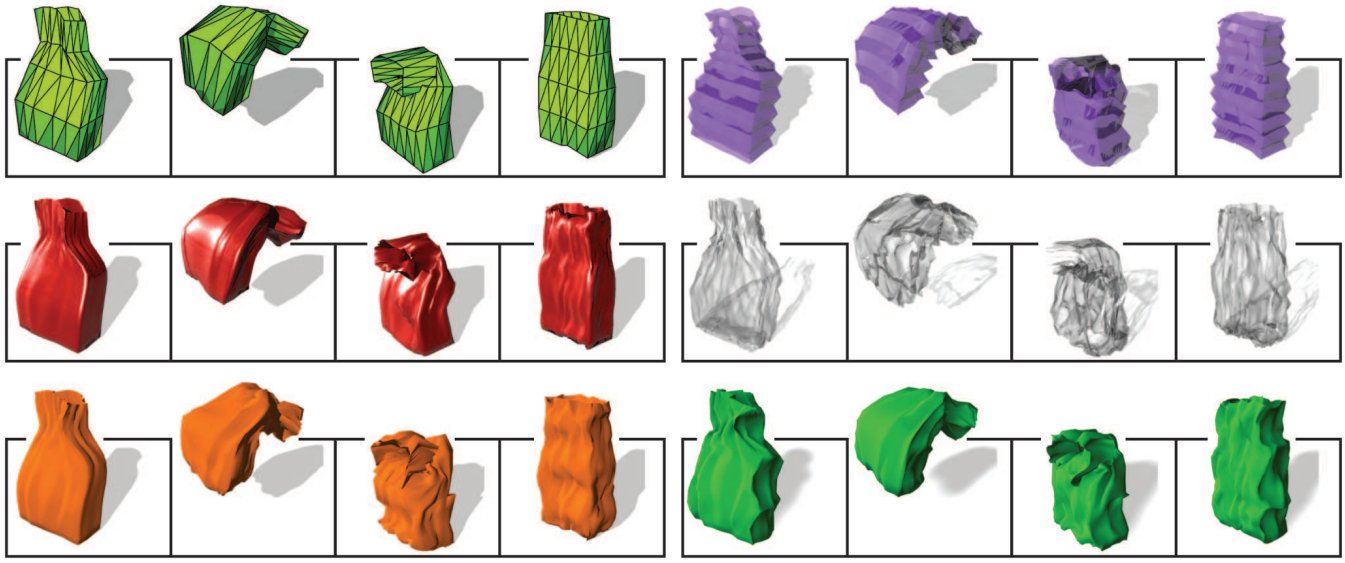


Figure 9: *Reusing motion-capture trajectories.* We naïvely rig a coarse mesh and use off-the-shelf motion-capture data to obtain the animation shown *top-left*, which, while riddled with self-collisions, serves as input to the automatic tracking process. The following sequences depict snapshots of a collision-free 1297-vertex mesh tracking the coarse motion under a variety of dynamical and material parameters.

the mocap sequence to a coarse (88 vertices) triangle mesh using an industry-standard rigging tool [Autodesk]. To create the initial fine mesh (1297 vertices), we twice subdivide the initial coarse mesh. Once again, we use the subdivision basis functions of the coarse mesh as the test functions for the weak constraints.

Holding the input trajectory fixed, we run the tracking algorithm using various dynamical and material parameters, obtaining a diverse range of results (refer to video and Fig. 9). The coarse mocap-induced motion is riddled with surface self-intersections, which are fully resolved during tracking.

Elasticity The first animation depicts a red vinyl material. Notice the fine vertically-oriented wrinkles and folds formed as the top of the bag contracts in the first snapshot, the crinkly top and boxy backside corners in the third snapshot, and the vertically-rippled appearance due to slight lateral contraction in the final snapshot. We achieve this look using a highly damped elastic shell model. Observe that damping affects only the material, *not* the coarse motion.

Damping The second animation depicts a flowy, undamped material reminiscent of an orange silk satchel. Because we use identical elastic parameters, in the first snapshot the red vinyl and orange flowy materials appear to be similar. However, since the second material is relatively undamped, its appearance diverges from the first as time advances. The impact of the backflip's hard landing (third snapshot) causes flowy, larger-wavelength horizontal waves (subtly, the smaller-wavelength vertical ripples due to the bag's lateral compression remain discernable).

Undeformed configurations The next pair of animations use our ability to track the coarse mesh using differing undeformed configurations. Applying coarse horizontal folds effects an accordion-like geometry on a purple bag; likewise, we obtain the appearance typical of a woven red-ribbon gift bag with a set of fine vertical ridges (refer to video). Altering the undeformed shape affects the objects' silhouettes and influences the formation and positioning of additional wrinkles, buckles, and folds (compare snapshot four to materials with a flat rest state)—these effects exceed what is achievable purely during rendering.

Complex collisions Building on this, we perturb the undeformed configuration with noisy offsets in the normal direction, achieving the appearance of a crinkled cellophane bag. The rough surface features did not noticeably alter runtime: tracking is affected by collisions much like an ordinary simulation; in contrast, many optimization methods rely on smoothness properties that are annihilated by rough collision landscapes.

External forces The final example incorporates a strong external wind force. This force excites the material without disturbing the gross motion, creating a psychedelic effect.

Unlike the coarse input, the tracked output is free of self-intersections. While one could conceivably attempt these material effects by subdividing the coarse motion and procedurally superimposing crinkly geometry or flowy motion, a simulation-based approach excels at producing temporally-evolving wrinkles and folds while steering clear of collisions present in the input data.

5.4 Physically detailed characters

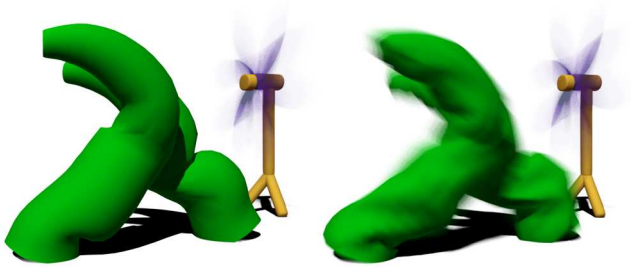


Figure 10: *Synergy of art and science.* We underscore the animator's broad gestures (*left*) with a simulated headwind (*right*). For equal comparison, both are rendered using 16-frame motion blur.

Our final example explores the role of *TRACKS* as a catalyst for synergy between art and physics. The artist models and animates

Xavier, an X-shaped biped (see Figs. 1,6,10) made of thin, flexible material. In doing so, she disregards physical details, such as wrinkling and buckling, and focuses on expressive motion, such as shifting body weight, sudden movement, and keeping time to the music. The tracking process enables us to outfit Xavier's motion with different materials and physical effects. As Xavier walks past his friends, observe the perfect roll-through of his feet—painting fine test functions on the feet enables us to precisely echo the artist's intent. Next, Xavier encounters a strong headwind (see Fig. 10). We underscore Xavier's struggle by introducing wind forces that make his body rapidly flutter. This would have been painstaking to achieve by hand. In the next scene, Xavier steps on a live wire, and by altering the undeformed configuration between a crinkly and smooth state, we (regrettably) simulate the consequent electrocution (see Fig. 1). Fortunately, Xavier "shakes off" the remaining charge and returns to his unruffled state.

5.5 Timing comparisons

	# verts (guide)	# verts (tracked)	# constraints	increase in DOFs	time per frame (s)	increase in time
§5.1	35	1625	4	0.2%	17.7 (17.0)	4.1%
§5.2	8	1538	8	0.5%	15.8 (14.9)	6.0%
§5.3	88	1297	88	6.8%	16.1 (15.7)	2.5%
§5.4	3038	3038	280	9.2%	43.2 (39.0)	10.8%

We report the runtimes on a 2GHz CPU for the various simulations throughout this paper using our tracking solver. For comparison, we have also listed in parentheses the runtimes obtained by simply running a standard simulation using the same parameters but with no tracking. We see a typical increase in simulation time of 5%–10% for the problems considered.

The exact runtimes we report for the examples in this paper are very specific to our implementation; however, our experience for all the example applications is that solving the constrained system incurs typically a 5%–10% overhead over an unconstrained simulation. Unlike collision detection for example, tracking is not done as a separate pass but as part of the simulation itself, so optimizing the simulation code would not change the percentage increase in runtime due to tracking.

6 Discussion

We presented *TRACKS*, a framework which enriches given input motion of thin flexible surfaces with physically simulated detail. Our approach may be viewed as a method inspired by the "line of action" metaphor [Thomas and Johnston 1981]—adding detail to a preview in a coarse-to-fine workflow. Our solver, in a single pass, provides such detail by integrating the equations of motion subject to constraints dictated by the input guide trajectory; we perform this integration using the framework of constrained Lagrangian mechanics. The scale at which our solver adds detail is controlled by the size and shape of Petrov-Galerkin test functions. We have shown the influence of the support and spatial distribution of test functions in controlling the characteristic shape of wrinkles and folds of thin shell materials. One particular avenue we plan to pursue in the future is to explore the effect of temporally varying test functions for input meshes that exhibit subtly changing detail over time.

In our current formulation of tracking, constraint forces always overpower the material's internal forces, without discrimination as to whether they arise from stretching versus bending response. Consequently, if the artist animates the mesh in a stretching mode, the underlying material will also stretch. While this is often desirable, there are cases where it is preferred that the material remain inextensible, even if the consequence is violation of the averaged constraints. Therefore, we are interested in extending the tracking formulation with a form of constraint regulation that effectively bounds constraint forces so that, when desired, they affect bending but not stretching modes. Furthermore, in the future, we would like to explore methods for constraining the *silhouettes* of animated characters, since art-directing the silhouettes is commonplace in traditional animation [Thomas and Johnston 1981].

Tracking offers the opportunity to experiment with a diverse range of thin shell appearances that defy emulation via geometric subdivision or per-frame post-processing. While our examples focused on thin plates and shells, the theory behind *TRACKS* requires only the existence of a computationally-tractable Lagrangian formulation, *e.g.*, such as those known for fluids and solid elastica. Therefore, we expect that this general framework also find applications in animation of smoke, liquids, biological tissue, and solid elastica. We hope that this paper will spur exploration in these and other directions.

Acknowledgments We would like to thank Rasmus Tamstorf, Brandon Michael Arrington, and Mathieu Desbrun for their influence on this project; Peter Schröder, Jerrold E. Marsden, Eva Kanso, and Denis Zorin for their valuable feedback; Kori Valz, Richard Giantisco, Max Min Weng, and Chris Bregler for their artistic contributions; and David Harmon for providing robust collision detection and response code. This work was supported in part by the DFG Research Center MATHEON in Berlin, the NSF (MSPA Award No. IIS-05-28402, CSR Award No. CNS-06-14770, CAREER Award No. CCF-06-43268), Autodesk, Disney Feature Animation, Elsevier, mental images, and nVidia.

References

ANGELIDIS, A., NEYRET, F., SINGH, K., AND NOWROUZEZAHRAI, D. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *SCA '06*, 25–32.

AUTODESK. Autodesk Maya Unlimited 8.0.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. of SIGGRAPH '98*, 43–54.

BOXERMAN, E., AND ASCHER, U. 2004. Decomposing cloth. In *SCA '04*, 153–161.

BRIDSON, R., FEDKIW, R. P., AND ANDERSON, J. 2002. Robust treatment of collisions, contact, and friction for cloth animation. *ACM TOG 21*, 3 (July), 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *SCA '03*, 28–36.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM TOG 21*, 3 (July), 586–593.

CAPELL, S., BURKHART, M., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2005. Physically based rigging for deformable characters. In *SCA '05*, 301–310.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM TOG 23*, 3 (Aug.), 905–914.

COHEN, M. F. 1992. Interactive spacetime control for animation. In *Proc. of SIGGRAPH '92*, 293–302.

CORDIER, F., AND MAGNENAT-THALMANN, N. 2005. A data-driven approach for real-time clothes simulation. *CGF 24*, 2 (jun), 173–183.

CUTLER, L. D., GERSHBEIN, R., WANG, X. C., CURTIS, C., MAIGRET, E., PRASSO, L., AND FARSON, P. 2005. An art-directed wrinkle system for CG character clothing. In *SCA '05*, 117–126.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. *ACM TOG 23*, 3 (Aug.), 441–448.

GALOPPO, N., OTADUY, M. A., MECKLENBURG, P., GROSS, M., AND LIN, M. C. 2006. Fast simulation of deformable models in contact using dynamic deformation textures. In *SCA '06*, 73–82.

GINGOLD, Y., SECORD, A., HAN, J., GRINSPUN, E., AND ZORIN, D. 2004. Simulating fracture and tearing of thin shells. Tech. rep., NYU.

GLEICHER, M., SHIN, H. J., KOVAR, L., AND JEPSEN, A. 2003. Snap-together motion: assembling run-time animations. In *2003 ACM Symposium on Interactive 3D Graphics*, 181–188.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: a simple framework for adaptive simulation. *ACM TOG 21*, 3 (July), 281–290.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *SCA '03*, 62–67.

GRINSPUN, E., Ed. 2006. *Discrete differential geometry: an applied introduction*. Course Notes. ACM SIGGRAPH '06.

HADAP, S., BANGARTER, E., VOLINO, P., AND MAGNENAT-THALMANN, N. 1999. Animating wrinkles on clothes. In *IEEE Viz. '99*, 175–182.

HAIRER, E., LUBICH, C., AND WANNER, G. 2006. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics. Springer.

HAUTH, M., ETZMUSS, O., AND STRASSER, W. 2003. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7-8, 581–600.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *SCA '04*, 131–140.

KIRCHER, S., AND GARLAND, M. 2006. Editing arbitrarily deforming surface animations. *ACM TOG 25*, 3 (jul), 1098–1107.

KONDO, R., KANAI, T., AND ANJYO, K. 2005. Directable animation of elastic objects. In *SCA '05*, 127–134.

LANCZOS, C. 1986. *The Variational Principles of Mechanics*, fourth ed. Dover.

LOVISCACH, J. 2006. Wrinkling coarse meshes on the GPU. *CGF 25*, 3 (Sept.), 467–476.

MA, L., HU, J., AND BACIU, G. 2006. Generating seams and wrinkles for virtual clothing. In *ACM VRCIA '06*, 205–211.

MALVERN, L. E. 1969. *Introduction to the Mechanics of a Continuous Medium*. Prentice-Hall, Englewood Cliffs, NJ.

MARSDEN, J. E., AND RATIU, T. 1994. *Introduction to Mechanics and Symmetry*, second ed., vol. 17 of *Texts in Applied Mathematics*. Springer-Verlag.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM TOG 23*, 3 (Aug.), 449–456.

NG, H. N., AND GRIMSDALE, R. L. 1996. Computer graphics techniques for modeling cloth. *IEEE CG&A 16*, 5 (Sept.), 28–41.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. *ACM TOG 21*, 3 (July), 291–294.

PARK, S. I., AND HODGINS, J. K. 2006. Capturing and animating skin deformation in human motion. *ACM TOG 25*, 3 (July), 881–889.

PLATT, J. C., AND BARR, A. H. 1988. Constraint methods for flexible models. In *Proc. of SIGGRAPH '88*, 279–288.

POPOVIĆ, J., SEITZ, S., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proc. of SIGGRAPH '00*, 209–218.

POPOVIĆ, J., SEITZ, S. M., AND ERDMANN, M. 2003. Motion sketching for control of rigid-body simulations. *ACM TOG 22*, 4 (Oct.), 1034–1054.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. 2004. Directable photorealistic liquids. In *SCA '04*, 193–202.

SCHENK, O., AND GÄRTNER, K. 2006. On fast factorization pivoting methods for symmetric indefinite systems. *Elec. Trans. Numer. Anal.*, 158–179.

SHI, L., AND YU, Y. 2005. Controllable smoke animation with guiding objects. *ACM TOG 24*, 1 (Jan.), 140–164.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM TOG 24*, 3 (Aug.), 417–425.

SINGH, K., AND KOKKEVIS, E. 2000. Skinning characters using surface oriented free-form deformations. In *GI '00*, 35–42.

SMITH, J., WITKIN, A., AND BARAFF, D. 2001. Fast and controllable simulation of the shattering of brittle objects. *CGF 20*, 2, 81–91.

STRANG, G., AND FIX, G. 1973. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM TOG 23*, 3 (Aug.), 399–405.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Proc. of SIGGRAPH '88*, 269–278.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proc. of SIGGRAPH '87*, 205–214.

THOMAS, F., AND JOHNSTON, O. 1981. *The Illusion of Life: Disney Animation*. Hyperion Books, New York.

THOMASZEWSKI, B., AND WACKER, M. 2006. Bending Models for Thin Flexible Objects. In *WSCG Short Comm.*

THÜREY, N., KEISER, R., PAULY, M., AND RÜDE, U. 2006. Detail-preserving fluid control. In *SCA '06*, 7–15.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM TOG 22*, 3 (July), 716–723.

WICKE, M., STEINEMANN, D., AND GROSS, M. 2005. Efficient animation of point-sampled thin shells. *CGF 24*, 3 (Sept.), 667–676.

WITKIN, A., AND BARAFF, D., Eds. 2001. *Physically Based Modeling: Principles and Practice*. Course Notes. ACM SIGGRAPH '01.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Proc. of SIGGRAPH '88*, 159–168.

WITKIN, A., AND WELCH, W. 1990. Fast animation and control of nonrigid structures. In *Proc. of SIGGRAPH '90*, 243–252.

WOJTAN, C., MUCHA, P. J., AND TURK, G. 2006. Keyframe control of complex particle systems using the adjoint method. In *SCA '06*, 15–24.

ZIENKIEWICZ, O. C., AND TAYLOR, R. L. 2000. *The finite element method*, fifth ed., vol. 1 and 2. Butterworth-Heinemann.

ZORDAN, V. B., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. In *SCA '02*, 89–96.

ZORIN, D., AND SCHRÖDER, P., Eds. 1998. *Subdivision for Modeling and Animation*. Course Notes. ACM SIGGRAPH '98.