HAL
open science

Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles

Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, Sheffer Alla

▶ **To cite this version:**

Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, Sheffer Alla. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles. ACM Transactions on Graphics, 2010, 29 (5), pp.157. 10.1145/1882261.1866183 . hal-00516411v3

HAL Id: hal-00516411

https://hal.science/hal-00516411v3

Submitted on 25 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles

Damien Rohmer[1,2], Tiberiu Popa[3], Marie-Paule Cani[1,2], Stefanie Hahmann[1], Alla Sheffer[4]

[1]Laboratoire Jean-Kutzmann, Université de Grenoble. [2] INRIA. [3]ETH Zurich. [4]University of British Columbia.

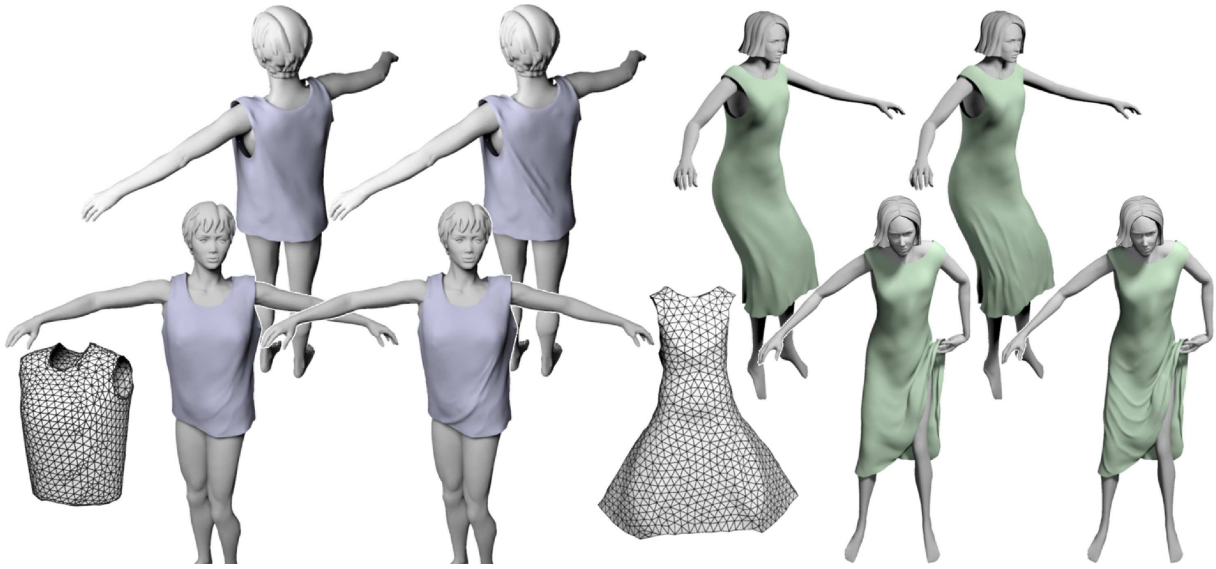


Figure 1: *Animation wrinkling (left to right): rest-shape mesh, frames from coarse simulation, and augmented with dynamic wrinkles.*

Abstract

Moving garments and other cloth objects exhibit dynamic, complex wrinkles. Generating such wrinkles in a virtual environment currently requires either a time-consuming manual design process, or a computationally expensive simulation, often combined with accurate parameter-tuning requiring specialized animator skills. Our work presents an alternative approach for wrinkle generation which combines coarse cloth animation with a post-processing step for efficient generation of realistic-looking fine dynamic wrinkles. Our method uses the stretch tensor of the coarse animation output as a guide for wrinkle placement. To ensure temporal coherence, the placement mechanism uses a space-time approach allowing not only for smooth wrinkle appearance and disappearance, but also for wrinkle motion, splitting, and merging over time. Our method generates believable wrinkle geometry using specialized curve-based implicit deformers. The method is fully automatic and has a single user control parameter that enables the user to mimic different fabrics.

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

Keywords: Cloth, wrinkle modeling, procedural animation, implicit modeling, mesh deformation

1 Introduction

Despite decades of research, efficient modeling of realistic cloth deformation remains a challenge. Although physically based simulation works well for computing dynamic effects and handling collisions, accurate simulation of folds and wrinkles is computationally expensive. This computational cost is particularly problematic in production settings, where users end up rerunning the simulation multiple times with different parameters to achieve the desired look.

An alternative, used in some production settings [Cutler et al. 2005], is to augment a coarse simulation using a procedural wrinkle model, in which the wrinkles are designed manually. This approach has two main drawbacks. First, designing wrinkle patterns manually is time consuming and requires specific skills. In addition, manual setting can only be done for specific frames, whereas real cloth wrinkles move continuously and deform over time, often exhibiting complex behaviors such as dynamic merging/separation.

Our method advances the augmentation idea one step further, introducing an automated post-processing step for believable wrinkle generation, to replace the manual intervention. The method can be applied on top of any pre-existing cloth animation, including soft spring animation or skinning. We observe that wrinkling behavior is strongly linked to cloth developability, or quasi-isometry between the deforming cloth and the corresponding 2D patterns or rest-shape [Popa et al. 2009]. Thus, given a coarse simulation output we use the stretch of the deformed surface with respect to this rest-shape as a guide for wrinkle placement. To obtain time-coherent animated wrinkling, wrinkle locations and magnitude are changed smoothly over time. The actual wrinkle geometry is generated using a novel implicit surface deformer mechanism which supports complex wrinkle shapes, including variations in radii and depth between and along wrinkles, as well as seamless transitions between adjacent and merging wrinkles (Figures 1,10).

1.1 Previous Work

Realistic cloth simulation has received considerable attention in the computer graphics community [Choi and Ko 2005]. Many of the issues addressed, such as collision processing, are outside the scope of this work, which focuses on realistic modeling of static and dynamic cloth fold and wrinkles. Previous methods addressing this problem can be grouped into three main categories.

Several methods add isometry-related constraints directly to a physically-based cloth model. Standard elastic models are ill-conditioned when limited compression takes place along the cloth surface, as shown in [Choi and Ko 2002]. Therefore using a spring model with increased stiffness to enforce isometry is problematic. Instead, [English and Bridson 2008] propose to use non-conforming elements to allow more degrees of freedom while minimizing triangle stretch during the simulation. [Thomaszewski et al. 2009] define the stretch constraint directly in the continuum model. These methods model various types of cloth with a high degree of accuracy and achieve impressive results. However, creating believable results requires finely tuned physical parameters and the methods are computationally very expensive (several minutes to more than one hour per second of simulation). This restricts their use to non-interactive, high-quality animations. [Muller and Chentanez 2010] use a second solver for adding wrinkles on a globally subdivided mesh. While the approach may be used in an automatic way, the subdivided edges may not be aligned with the wrinkles leading to visual artifacts.

Other approaches facilitate the design of realistic cloth shapes by minimizing energy functions that measure developability. Though not fully physically-based, such formulations have the advantage of allowing the specification of positional constraints. [Tang and Chen 2009] approximate a set of input points with a quasi-developable surface, enabling static cloth design. [Popa et al. 2009] use a weighted space-time optimization to generate coherent wrinkles over time on capture-generated cloth surfaces. They use the video input from the capture to obtain wrinkle locations.

A fully geometric approach for wrinkling garments is proposed by [Decaudin et al. 2006]. They use prior knowledge of cloth buckling behavior when wrapped around cylindrical body parts to generate pre-defined types of procedural folds. The method is restricted to the case of cylindrical cloth surfaces experiencing a small set of pre-defined deformations (compression, twisting or folding along the cylinder's axis) and cannot be combined with arbitrary cloth animation.

A number of methods have been proposed to efficiently generate procedural wrinkles on top of coarse cloth animations. [Hadap et al. 1999] compute a compression intensity map over the surface. The map is subsequently used to locally weigh a pre-painted texture pattern modeling the wrinkles. The weighting is computed such that the area of compressed triangles is approximately restored. The resulting texture is used as a bump-map for very fast wrinkle rendering. [Kimmerle et al. 2004] extend this idea to procedurally generated texture patterns, which are locally oriented in the triangle compression direction. The deformation is performed using displacement mapping. Both methods can be applied to animation by blending multilayered textures at the price of violating the area conservation constraints. However, time-varying fold orientation cannot be accurately modeled using texture interpolation. Moreover, in both settings wrinkle frequency and width have to be pre-defined by the user. We too use a procedural approach for wrinkling, but our method generates actual geometric wrinkles on the input surface and directly ensures wrinkle coherence over time.

[Cutler et al. 2005] define cloth wrinkles as curves on the input cloth surface and use those as deformers for generating the final geometry. The curve shapes and influence radii are manually designed for a number of specific frames, and the associated wrinkles fade in and out when intermediate frames are computed. This approach enables artists to create the desired fold shapes but requires significant user time and expert design skills. Our method uses a similar idea of encoding wrinkles as surface curves. However these curves are automatically positioned and animated over time by analyzing the coarse simulation output.

Recently, [Wang et al. 2010] and [de Aguiar et al. 2010] used a machine learning approach to generate real-time wrinkles, they require training data sets with similar examples which limit the range of application of the method.

1.2 Overview and Contributions

This paper presents a novel efficient procedural method to enhance a pre-existing coarse cloth animation with realistic-looking animated wrinkles. Fabric wrinkles typically form in areas where the surface is compressed to allow for preservation of material, and are nearly perpendicular to the direction of compression. Based on this observation our method analyzes the stretch tensor of the coarse simulation output when compared to the user-given rest-shape and places wrinkle curves in areas where the tensor indicates compression or shrinkage, tracing curves along streamlines of the tensor orthogonal to the main shrinkage direction (Section 2, Figure 2, left). In order to maintain temporal coherence between wrinkles in dynamic scenes, the method is not applied independently in each frame but uses the computed tensor to smoothly propagate existing curves from one frame to the next, and to add or delete curves where necessary (Section 3, Figure 2, middle). Real-life wrinkles form to improve isometry with respect to the cloth rest-shape with their width or radius linked to the fabric type and thickness. Our method uses the stretch tensor to determine local wrinkle width and depth, such that adding wrinkles reduces the surface stretch with respect to the rest-shape. Given the computed set of curves and the associated depth and width values the method generates believable wrinkle geometry using new curve-based implicit deformers (Section 4, Figure 2, right).

The combined system provides an effective, fast, and fully automatic mechanism to create realistic-looking, complex dynamic wrinkles augmenting the input animation. Our tensor-based wrinkle curve tracing places curves in locations consistent with the input animation, while the temporal propagation mechanism supports wrinkle motion and deformation over time, enabling wrinkles to slide along the cloth surface and to smoothly change shape and orientation following cues provided by the input animation sequence. By using the new implicit curve-based deformers we can model complex wrinkle shapes with smoothly varying width and depth and effortlessly support seamless wrinkle mergers. Finally, our computation of the wrinkle width and depth parameters reduces the stretch between the final cloth surface and the rest-shape, reflecting real-life wrinkle behavior, even though we do not aim to exactly preserve the isometry to the rest shape.

The user can mimic the impact of different fabrics on the wrinkle size using a single parameter $R_{\min}$ which defines the minimal wrinkle curvature radius (see Section 4.1).

2 Static wrinkle curve generation

This section describes the extraction of 1D curves representing wrinkles given a deformed cloth mesh and the associated rest-shape. The proposed method can be used as-is to add wrinkles to coarse simulation outputs for static settings, such as dressing a mannequin. In animation settings this process is combined with the curve propagation method described in Section 3 to achieve temporal continuity.

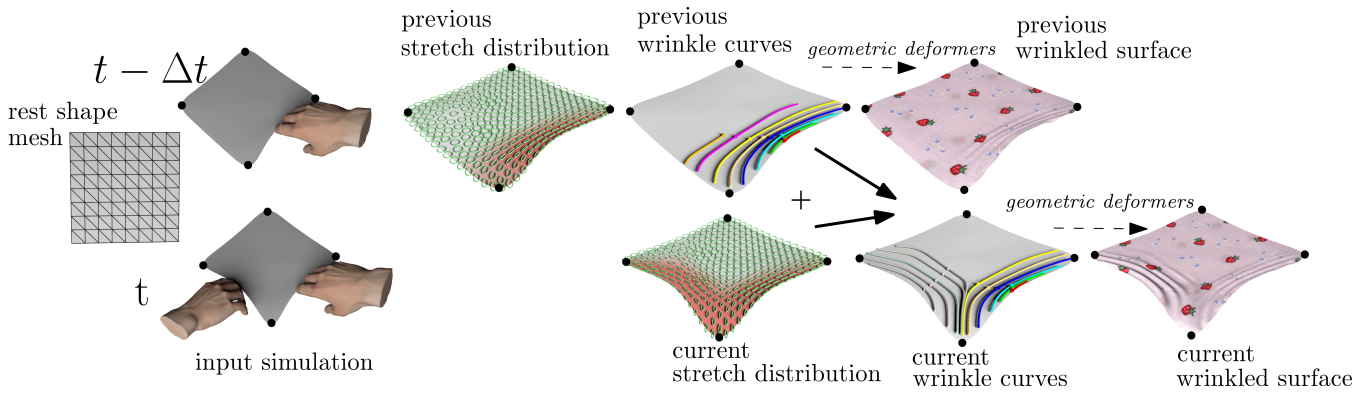


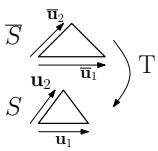
Figure 2: *Algorithm overview: The current stretch tensor field (visualized by ellipses scaled and oriented according to its eigenvalues and eigenvectors) and the previous wrinkle curves (indicating wrinkle orientation) are used to generate the current wrinkle curves while ensuring temporal coherence. Implicit deformers are used to generate the wrinkle geometry. The time-step length has been artificially increased for illustration purposes.*

In many cases, the rest-shape is a planar 2D pattern. If the rest-shape is not planar, we segment it into near-developable charts and parameterize these in the plane [Sheffer et al. 2007], in order to simplify subsequent curve tracing. We use the ABF++ parameterization method [Sheffer et al. 2005] to get a good trade-of between length and angle preservation. To maintain coherence along chart boundaries we constrain the parameterization to preserve boundary edge lengths [Sheffer et al. 2007]. We use the parameterization to define a common coordinate frame for the rest-shape triangles and to speed up the curve tracing.

2.1 Wrinkle vector field from stretch tensor

As noted earlier, cloth wrinkles occur in compressed regions and are orthogonal to the main shrinkage direction. We are thus looking for a *wrinkle vector field* in the cloth surface space whose magnitude reflects the amount of compression and whose direction is orthogonal to the local direction of compression. In computer graphics stretch is often measured using the formulation of [Sander et al. 2001]. However, we found that the following alternative stretch tensor definition used in continuum mechanics [Talpaert 2003] provides more consistent vector field directions.

Let S be the 3D mesh representing the deformed cloth and $\overline{S}$ the corresponding rest-shape. The 2D affine deformation from a triangle in $\overline{S}$ to its counterpart in S can be expressed by the 2x2 matrix T.



Let $(\mathbf{u}_1, \mathbf{u}_2)$ and $(\overline{\mathbf{u}}_1, \overline{\mathbf{u}}_2)$ be the 2D edge vectors representing the shape of the triangle in the local frames of S and $\overline{S}$ respectively. Then T is given by $\mathrm{T} = [\mathbf{u}_1, \mathbf{u}_2][\overline{\mathbf{u}}_1, \overline{\mathbf{u}}_2]^{-1}$. The deformation is isometric (zero stretch) iff $\mathrm{T}^T = \mathrm{T}^{-1}$. The 2×2 symmetric positive definite matrix $\mathrm{U} = \sqrt{\mathrm{T}^T\mathrm{T}}$, called the *stretch tensor* corresponding to the square root of the *Right Cauchy-Green deformation tensor* in continuum mechanics , is thus a good measure of compression and elongation. Assuming that there are no degenerate triangles in the mesh, U can be diagonalized as

$$\mathrm{U} = \lambda_1 \, \mathbf{e}_1 \mathbf{e}_1^T + \lambda_2 \, \mathbf{e}_2 \mathbf{e}_2^T \, , \tag{1}$$

where $\lambda_1 \leq \lambda_2$ are real, positive eigenvalues, and $\mathbf{e}_1, \mathbf{e}_2$ are unit eigenvectors, defining main stretch or shrinkage directions (Figures 2, 3).

To extract a wrinkle map from this per-triangle tensor field we have to express it as a smooth per-vertex map describing well-defined and smooth compression directions throughout.

Using the common coordinate frame, defined by the pre-computed planar parameterization, we define the stretch tensor $\mathbf{U}_i$ at a vertex i as: $\mathbf{U}_i = \bigoplus_{j \in \mathcal{V}_i} \omega_j \mathbf{U}_j$, where $\mathcal{V}_i$ denotes the index set of the triangles incident to vertex i, and $\omega_j = \mathcal{A}_j / \sum_{k \in \mathcal{V}_i} \mathcal{A}_k$ are triangle-area based weights. A simple component-wise averaging of the tensors dissipates much of the anisotropic information. Instead we use an interpolation scheme based on an affine-invariant Riemannian metric defined in tensor space [Pennec et al. 2006], with $\oplus$ being the iterative weighted means in the exponential map, where $\mathbf{U}_i$ are obtained by iteratively applying the following formula until convergence

$$\mathbf{U}_i = \mathbf{U}_i^{1/2} \exp\left(\sum_j \omega_j \log\left(\mathbf{U}_i^{-1/2} \mathbf{U}_j \mathbf{U}_i^{-1/2} \right) \right) \mathbf{U}_i^{1/2} \tag{2}$$

This scheme provides smoothly varying eigenvalues and, more importantly, smoothly rotating eigenvectors with little anisotropy dissipation. We define the continuous stretch tensor field $\mathbf{U}$ over the whole mesh using barycentric coordinates and the tensor weighting operator $\oplus$.

For wrinkling purposes, we are only interested in areas of the mesh where the coarse animation results in compression, i.e. locations where the smallest eigenvalue λ_1 is less than one. Hence we define a *wrinkle vector field* $\mathbf{v}$ at any point on the surface as

$$\mathbf{v} = \max(1 - \lambda_1, 0) \, \mathbf{e}_2 \, . \tag{3}$$

The intensity of $\mathbf{v}$ reflects the rate of shrinkage per unit of length wherever shrinkage occurs and its direction is orthogonal to the main shrinkage direction.

2.2 Tracing wrinkle curves

Our goal is to efficiently extract wrinkle curves that provide a plausible interpretation of the surface behavior in response to the measured compression. Thus we aim to compute curves that capture the shape and magnitude of the compression vector field, while providing a good sampling of compressed regions. To this end, one approach could be to trace curves in areas of high compression, while enforcing a minimal distance between them. However, real-life wrinkles can in fact merge or come very close together in some regions, violating such minimal distance constraints. Instead, to generate believable wrinkles we trace curves one-by-one starting from seed points with high compression magnitude, while constraining each new seed to be at a minimal distance from earlier traced curves. By enforcing minimal distance for seed points only we allow curves to come arbitrarily close-by. Our implicit deformer

method for generating wrinkle geometry (Section 4) robustly handles such merging behavior.

We define wrinkle curves as parametric curves $\bar{\gamma} : [0,1] \rightarrow \bar{S}$ extracted along streamlines of the wrinkle vector field $\mathbf{v}$ over the parameterized rest surface by integrating the 2D ordinary differential equation:

$$\begin{cases} \overline{\gamma}'(u) & = \pm \mathbf{v}(\overline{\gamma}(u)) \\ \overline{\gamma}(0) & = \overline{\mathbf{s}} \\ \|\mathbf{v}(\overline{\gamma}(u))\| & \geq 1 - \lambda_{\max} \,, \end{cases}$$

where $\overline{\mathbf{s}}$ is a starting seed position, and $(1-\lambda_{\max}) \in [0,1]$ is a compression threshold characterizing regions of sufficient compression. In practice we use $\lambda_{\max} = 0.9$.

To initiate the tracing we order the mesh vertices according to their compression rates placing them in a priority queue. Starting with the vertex with maximal compression as the first seed, we proceed as follows:

1. **Integration:** Since $\mathbf{v}$ is sufficiently smooth thanks to our continuous expression for the stretch tensor $\mathbf{U}$, we successfully trace a smooth curve starting from the seed using forward Euler integration. As $\mathbf{v}$ is non-oriented, we propagate the curve in both directions from the seed-point, checking that the dot product between the previous direction and the new one stays positive during the integration. The integration stops when the local compression rate $\|\mathbf{v}\|$ becomes smaller than $1 - \lambda_{\max}$, or when the surface boundary is reached.

2. **Seed Selection:** The next seed point is selected by extracting the next vertex from the queue while discarding those which are too close to the already computed wrinkle curves. The required minimal distance is set to twice the minimal radius of wrinkle curvature $R_{\min}$, as any two wrinkles lying at a closer distance than that would effectively overlap throughout.

The process terminates when the queue is empty.

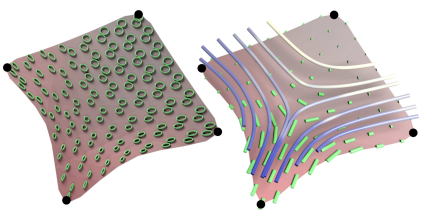


Figure 3: *Wrinkle tracing: stretch tensor (left); wrinkle vector field and wrinkle curves (right).*

The 2D curves $\overline{\gamma}$ are mapped to the deformed surface, with the resulting curves γ indicating the loci and directions where wrinkle geometry should be added to compensate for local shrinkage (Figure 3). The compression rate along each curve is used to control the shape of the wrinkle as discussed in Section 4.

3 Time coherent wrinkle curve animation

Real cloth wrinkles are continuous over time. However, using the algorithm in Section 2 as-is to independently generate wrinkles at each frame of an animation sequence can lead to inconsistent sets of wrinkles popping in and out. [Popa et al. 2009] relied on input video combined with fairly costly, space-time optimization to achieve the temporal continuity. We do not have the video prior and aim for a very fast solution to achieve the continuity. To this end we use a combination of particle-based wrinkle-seed propagation and temporal smoothing to achieve the desired effect.

Particle-based wrinkle-seed propagation: We observe that due to the smoothness of the input animation the wrinkle vector field changes gradually from one frame to the next. We thus expect many of the folds in the previous frame to remain in the current one with possible small position or shape changes. To achieve this wrinkle sliding effect, we initialize the wrinkle curves for each frame with those in the previous one. Recall that our static processing placed wrinkles by first placing wrinkle seeds at local compression maxima and then tracing them along the vector field streamlines. To extend this process to the dynamic setting, we aim to attract the wrinkle curves toward the new local maxima of compression. For efficiency we do not search for local maxima along the entire curve but only for the original seed point. To attract the seed to a local maximum we use a particle-based animation setting, as follows.

The particle-based animation processes seeds in an order based on the amount of compression, given by the wrinkle vector field at the current frame. For each seed $\overline{\mathbf{s}}$ previously positioned at $\overline{\mathbf{s}}(t)$ at time t, its new position at time $t + \mathrm{d}t$ is expressed as $\overline{\mathbf{s}}(t) + \mathrm{d}\overline{\mathbf{p}}$, where $\mathrm{d}\overline{\mathbf{p}}$ is a small displacement vector on the 2D pattern modeling the movement of the wrinkle. We use a non-zero $\mathrm{d}\overline{\mathbf{p}}$ if the compression at the seed-point location decreased from the previous frame. In this case, the displacement vector $\mathrm{d}\mathbf{p}$ is oriented in the direction of the gradient of greatest compression $\nabla\|\mathbf{v}\|$ in order to move towards the closest local maximum of compression. To find the optimal displacement we do a line search along the gradient direction, bounding the maximal displacement length allowed to $R_{\min}$. This choice prevents wrinkles from sliding more than half their width in one step, leading to visually natural-looking results.

To avoid multiple wrinkles from converging toward the same local maximum, we force each seed to stay at a distance of at least $2 \times R_{\min}$ from previously computed wrinkle curves. We then retrace the curves using streamline integration (Section 2.2). At the extremities of each wrinkle we check the amount of displacement with respect to previous position, and shorten the curve if the displacement is larger than $R_{\min}$.

Adding and Deleting Curves: We add to this animation process a mechanism to delete vanishing wrinkles and to create new ones.

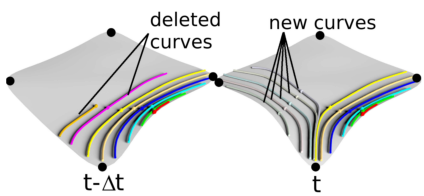


Figure 4: *Wrinkle curve propagation between frames. Curve seeds are indicated by spheres. Several curves remain in place or slide, two curves are deleted, and five new ones are added in the newly compressed area.*

If the local compression is not sufficient after the allowed amount of sliding for the wrinkle seed, the seed and the associated curve are deleted for the subsequent frames. Finally, after every previously existing seed is processed, new wrinkles are generated in newly compressed regions by applying the same procedure as in the static case.

Temporal smoothing: The algorithm we just described provides a plausible approximation of wrinkle behavior over time. In particular, the local amplitude of compression increases and decreases smoothly enough, so we observed no visual popping when wrinkles are inserted of deleted. However, although we generate continuous trajectories for seed points, the integration of wrinkle curves is computed independently at each time step. Hence the trajectory of points far from the seed may not be continuous enough.

To achieve temporal continuity we smooth the integrated trajecto-

ries in the time domain. Let the wrinkle curves be consistently parameterized is space by a parameter u. Then a given wrinkle curve can be considered as a space time function $\overline{\gamma}(u,t)$. We apply a 1D low-pass filter on each time dependent curve $\overline{\gamma}(u = \text{const}, t)$ in order to ensure smooth transitions between frames.

4 Wrinkle geometry

Once the wrinkle curves are computed for all the animation frames, the final step is to generate the actual, realistic-looking wrinkle geometry at each frame. Real-life wrinkles have varying widths and depths reflecting a combination of the amount of compression involved and the fabric's thickness and structure. For instance, compressing silk can generate very fine wrinkles while compressed leather or felt form much coarser folds.

4.1 Wrinkle Parameters

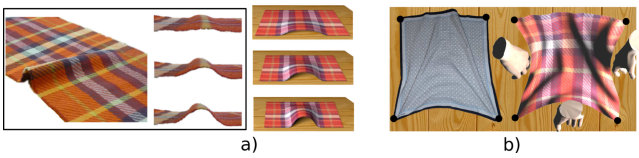


Figure 5: *a: Measurement of curvature evolution with respect to a given compression rate on real cloth (photos at left), and a progressive wrinkle modeled with our approach (right). b: Wrinkle merging on real cloth (left) and using our method (right).*

To control wrinkle shape we use two parameters affecting its width and depth: the curvature radius $R(u)$ along each wrinkle curve, and the wrinkle offset $\beta(u)$ (Figure 6, left), determining the portion of the circular arc used for forming the wrinkle. We use the notion of minimal radius of curvature R_{min} to account for the type of fabric. The radius is set by the user and in our experiments provides intuitive control of wrinkle behavior. The actual wrinkle radii are a function of this limit and of the compression rate computed from the wrinkle vector field, with the wrinkle radius decreasing as the compression increases. We are not aware of any existing physical law or empirical study on this topic. We therefore conducted our own experiments on various samples of real cloth by measuring curvature radii and comparing them to the compression rate computed in a corresponding simulation. See Figure 5 (a) for the example of a wool scarf. Based on those experiments, we derived the following relationship:

$$R(u) = \left(\frac{1 - 2/\pi}{v(u)}\right) R_{\text{min}} \,,$$

where $v(u) = \|\mathbf{v}(u)\|$ is the compression rate, i.e. the norm of the wrinkle vector field along the wrinkle curve, computed using Equation (3). This equation interpolates the surface curvature along the wrinkle between infinity when no compression occurs, to $1/R_{\text{min}}$ when the compression corresponds to a half circular profile (when $v(u) = 1 - 2/\pi$).

Once the radius is set, the actual width (and related height) of the wrinkle should be such that that the shrinkage with respect to the rest shape is minimized. We express those using the offset β which is subsequently used by our implicit deformer. The offset β is computed so that the increase in length Δl due to the wrinkle compensates for the local shrinkage of the mesh. All the relationships needed for this computation are summarized in Figure 6, left.

4.2 Implicit Deformers

We aim to create wrinkle geometry that follows the previously computed curves and conforms to the computed local wrinkle parameters. We also aim to smoothly handle close-by wrinkles, avoiding

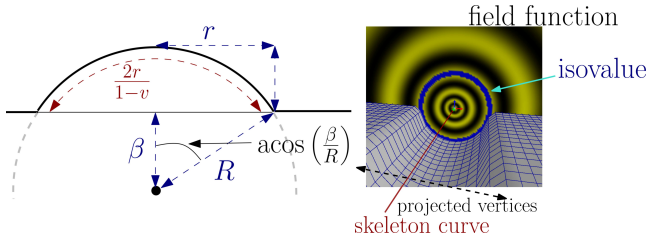


Figure 6: *Left: Relationship between the width r, the height h, and the arc-length l of a wrinkle, generated using a given offset distance β. Right: the projection step performed to create the wrinkle profile. Mesh vertices inside the region where $f > 1$ are projected onto the isosurface.*

self-intersections and other artefacts. Previous procedural settings [Cutler et al. 2005; Decaudin et al. 2006] used pre-defined or manually designed wrinkle shapes, and thus cannot be used for more general wrinkles. The deformation approach of [Popa et al. 2009] can potentially handle diverse wrinkle parameters, but has no obvious way to process close-by and overlapping wrinkles. Instead, our method relies on new implicit deformers specifically designed to have the desired specifications. The implicit formulation supports smoothly varying wrinkle parameters, robustly handles complex wrinkle interactions such as wrinkles smoothly merging or splitting and prevents collisions between adjacent wrinkles by implicitly replacing intersection with merging. Intersection between wrinkled geometry and the character's mesh are prevented during the same deformation step as discussed below.

Recall that an implicit surface (see [Bloomenthal 1997] for details) is an iso-surface $f(\mathbf{p})$ of a smooth field function f, often defined as a decreasing function of the distance to a source element, referred to as a skeleton. In the definitions below the iso-value used to define all surfaces is one. Implicit surfaces are known for their ability to smoothly blend when they are close-by by summing up the field contributions from the different skeletons. We define the implicit geometry of the actual wrinkles *by using the wrinkle curves as the skeletons*. The resulting implicit generalized cylinders are then blended together, modeling the smooth merging and splitting behaviors. The resulting structures are used as *deformers* to locally deform the surface mesh around them, by projecting neighboring mesh vertices to the combined iso-surface.

Wrinkle primitives: Using the convolution surface model to get bulge-free blends we define each implicit wrinkle primitive as the integral of a kernel function along the skeleton curve $c(u)$:

$$f(\mathbf{p}) = \int_u \omega(u)\, \kappa(\|c(u) - \mathbf{p}\|)\, \mathrm{d}u \,, \tag{4}$$

where $\omega(u)$ is a weighting function controlling the radius of the deformer and κ is a kernel function aligning it with the wrinkle curve. In this paper, we use the Cauchy kernel $\kappa(x) = 1/(1 + s^2x^2)^2$, with s set to one hundred, for meshes scaled to fit into the unit cube, to reduce the distance at which implicit primitives start to blend and compute the convolution integral using a closed form solution [Mccormack and Sherstyuk 2001].

To obtain the desired deformer radius we analyze the analytical expression of the Cauchy integral. Due to the fast decay of the Cauchy kernel, the radius almost reaches a limit value that depends only on $\omega(u)$, except at the extremities of the primitive (see Figure 8 (right) for typical shapes generated with a constant $\omega(u)$). Using the formula giving this limit radius, we obtain $\omega(u) = 4s(1 + s^2R(u)^2)^{3/2}/\pi$. The offset parameter β (Section 4.1) is used to position the wrinkle skeleton at a varying offset distance

$\beta(u)$ below (or above) the associated wrinkle curves, enabling us to model shallow to deep wrinkles mimicking real examples such as those in Figure 5.

Surface deformation: We aim to blend the wrinkles with the actual underlying mesh surface for which we have no implicit definition. Computing such a representation, for instance a convolution surface based on the mesh triangles, is very costly. Instead we introduce a new approach for locally blending implicit and explicit geometry based on the observation that at any point a mesh can be locally approximated by its tangent plane.

We define an extra implicit primitive, called the *mesh-deformer*, whose contribution at each point $\mathbf{p}$ is set equal to the contribution of the tangent plane of the closest point on the mesh. Calling z, the distance between $\mathbf{p}$ and the closest point:

$$f_{\text{mesh}}(\mathbf{p}) = f_{\text{plane}}(\mathbf{p}) = \frac{\pi}{s^2(1 + s^2(z(\mathbf{p}) + z_0)^2)} \; .$$

with $z_0 = 1/s\sqrt{\pi/s^2 - 1}$ used to enforce the isovalue 1 to be located on the mesh.

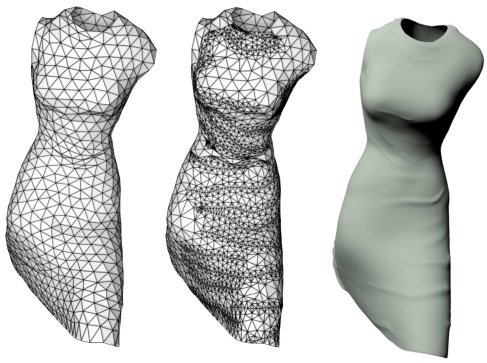


Figure 7: *Wrinkle Geometry: dress mesh before and after refinement, and final geometry*

The actual deformation consists of locally projecting the mesh vertices along their normal direction onto the iso-surface generated by summing the field contributions of the mesh-deformer and those of the wrinkle primitives, as shown in Figure 6 (right) for a single wrinkle. To prevent wrinkle-deformers from affecting several mesh layers when the mesh represents folded cloth, their contribution is only applied to the surface in the vicinity of the associated wrinkle curve, up to a distance where their influence can be neglected ($2 \times R(u)$ with our setting of the Cauchy kernel). To accurately capture wrinkle geometry, the mesh is first locally refined in the region of influence of the wrinkle curves. The curves are embedded into the refined mesh to align mesh edges with the wrinkle top. The refined mesh is generated using constrained Delaunay triangulation [Shewchuk 2002]. The resulting refined mesh (Figure 7), smoothly joins with the coarse triangulation.

Note that using a projection scheme is necessary, since simply summing displacement fields associated with skeleton curves, as in [Singh and Fiume 1998], would generate visible artifacts in our case (Figure 8). Instead, our scheme takes advantage of the almost bulge-free behavior of convolution surfaces when curve primitives joint. The combined action of the wrinkle and mesh deformers is computed at interactive rates and generates believable wrinkle shapes, reflecting the compression applied by the coarse animation and the fabric properties.

Wrinkling direction and collision avoidance: By choosing the appropriate offset and projection directions each wrinkle deformer

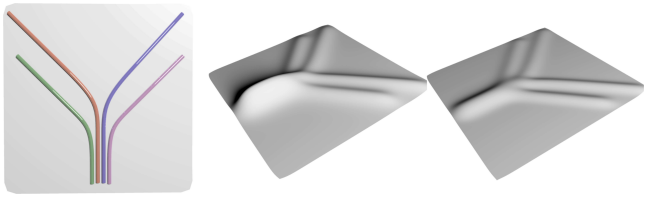


Figure 8: *Comparison between a deformation algorithm where vertices are moved according to the sum of displacement fields associated with individual wrinkles (middle), and our solution where vertices are projected to the iso-surface (right).*

can be used to control the direction of the wrinkles with respect to the mesh normal . For dressed characters we by default form outward pointing garment wrinkles thus avoiding creation of extra collisions between the deformed mesh and the character's body. More complex collisions situations with obstacles on both sides of the coarse mesh are handled by re-projecting the colliding wrinkle vertices onto the external object surface when applying the deformers, with the effect of locally suppressing wrinkles in these regions.

5 Results

We applied our algorithm to various cloth simulation scenarios and analyzed the results in terms of shrinkage reduction, visual realism, and computational time.

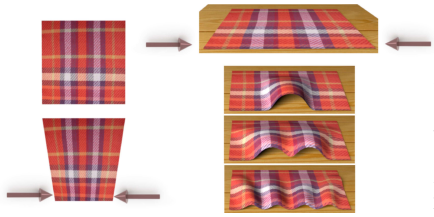


Figure 9: *Left: rest-shape (top) and deformed surface (bottom). Right: Three different cloth types compensating for the same amount of compression.*

Measurements: Figure 9 demonstrates simulation of the behavior of different types of cloth by changing the material constant R_{min}. It shows a planar piece of cloth undergoing unilateral compression on the front, with three different values of the minimal curvature radius resulting in different wrinkle magnitudes and frequencies. When defining the wrinkle size our method aims to reduce the shrinkage with respect to the planar patterns. To evaluate its impact we measured the before and after stretch for the examples in Figure 9. The coarse simulation output exhibits a maximal length error of 36% along the front. Our algorithm nearly restores the original length with just a 2% error. This remaining error is largely due to the blending of the implicit deformers with the mesh. In the opposite direction, .i.e. along the wrinkle curves, shrinkage stays below 5%. Finally, we measured the global stretch on the input and final meshes, expressed as an L^2 norm of the compression magnitudes. We obtained a 20% reduction in the global stretch, although restoring isometry to the rest-shape was not an explicit goal of our method.

Visual Realism: We performed qualitative comparisons of our results with real-life outputs for two scenarios: a piece of cloth pushed from several directions (Figure 5) and a towel pushed forward by a leg (Figure 11). As shown by these two examples our results look quite similar to the real-life ones. Clearly exact comparison is not possible as the inputs conditions were not identical. As shown in Figure 11 we also compared our result to a fine simulation achieving comparable results in a fraction of the time.

Figure 10: *Simulating different fabrics. From left to right: frame from a coarse simulation; two frames augmented with believable wrinkles for different types of fabric; zoom on merging wrinkles*

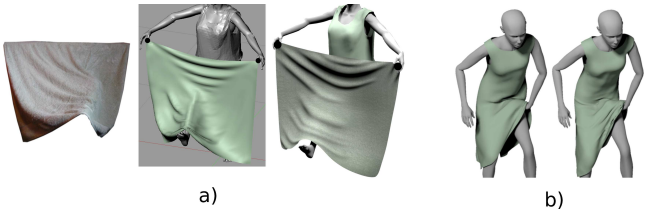


Figure 11: *While high resolution simulation takes 10 to 25s per frame, our method takes 2s per frame and exhibits even finer details. a: Real experiment (left), high resolution simulation (middle) and the output of our method (right). Note the consistency between our and real wrinkle orientations. b: High resolution simulation (left) and our wrinkles (right).*

Application to complex geometry: We tested our algorithm on a number of diverse garment animations. The coarse input simulations were created using Blender[1]. Figure 1 shows a dress and a t-shirt worn by an animated character. The wrinkles our method creates on both are consistent with the motion. As shown in the associated video, our approach generates a believable dynamic behavior for the wrinkling t-shirt when the character suddenly stops rotating. An example of using our method to add wrinkles to skinning based animation in shown in Figure 12, with realistic wrinkles showing up on the tights as the character's leg bends. Figure 13 shows three other interesting animations, including two dressed female characters with different dresses and motions and a dressed bunny. The corresponding complete sequences are shown in the accompanying video. In all the examples the added wrinkles visibly enhance the coarse inputs. These examples demonstrate that fast simulation automatically augmented with fine wrinkles provides a viable, believable alternative to full-blown simulation.

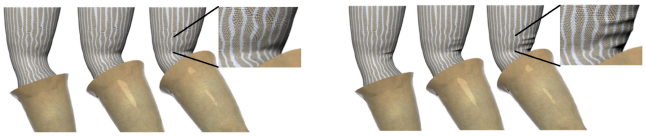


Figure 12: *Wrinkling skinning animation (left), note the gradually appearing wrinkles on the tights (right).*

Runtimes: We tabulated the runtimes of our method executed on a single CPU and compared them with the cloth simulation module available in the open-source Blender software. This cloth simulation module is based on implicit integration and uses the collision processing method from [Bridson et al. 2002].

In our experiments, low resolution simulations were computed at 25ms to 2s per frame depending on the collision complexity. Adding our wrinkles generally takes 1s to 2s per frame. To compare with, higher resolution simulations obtained after uniform mesh subdivision such that triangle edges are not longer than our minimal curvature radii run at 10 to 25 seconds per frame, so our method gains one order of magnitude. The following table summarizes our computational time in ms per frame. We separate wrinkle curve processing from geometry generation done by locally projecting vertices to the implicit surface.

	coarse vertices	wrinkle curve	projected vertices	projection time
Basic rectangle	400	100ms	475	118ms
Towel	80	50ms	300	75ms
T-shirt	1168	800ms	1500	750ms
Bunny's dress	364	300ms	1000	700ms
Tights	1112	400ms	500	400ms
Dress Fig. 13	931	700ms	900	600ms
Dress Teaser	1249	850ms	3500	2700ms

Note that the number of projected vertices depends mainly on the re-triangulation criteria and on the number of wrinkles. The projection algorithm is clearly the slowest part of our algorithm. Fortunately, while the wrinkle curves computation needs to be performed iteratively frame-to-frame, this more time-consuming step is independent for each frame and thus can be run in parallel.

6 Discussion and Conclusion

We have presented a procedural method that augments a pre-existing cloth animation with realistic-looking wrinkles at interactive rates. Being able to quickly enhance a coarse dynamic simulation with plausible visual details is an appealing approach when no accuracy is required. This paper advances this approach one step further by enabling the automatic creation of believable cloth wrinkles using cues provided by the coarse animation. The wrinkle placement is based on an analysis of the stretch created by the animation and the wrinkle shape and motion are continuous over time. Although limited by the lack of accurate information on the way real cloth wrinkles move and deform, our procedural model, tuned from a few experiments with real cloth, achieves believable results. Since it is used as a post-process, wrinkle generation can be tuned independently from the coarse simulation by adjusting the wrinkle radius parameter, simplifying user control.

One of the contributions of this work are the new implicit deformers used for wrinkle generation. Visually realistic wrinkle shapes with varying width and depth that smoothly merge and separate are generated thanks to the blending between curve-based primitives and our new mesh-deformer. The shapes we generate for wrinkles look quite believable and reduce the local compression, although they do not ensure exact isometry with the pattern, nor do they ensure that the created surface is developable. This was not our goal.

Although flat to rounded wrinkles can be represented by our

[1] http://www.blender.org/

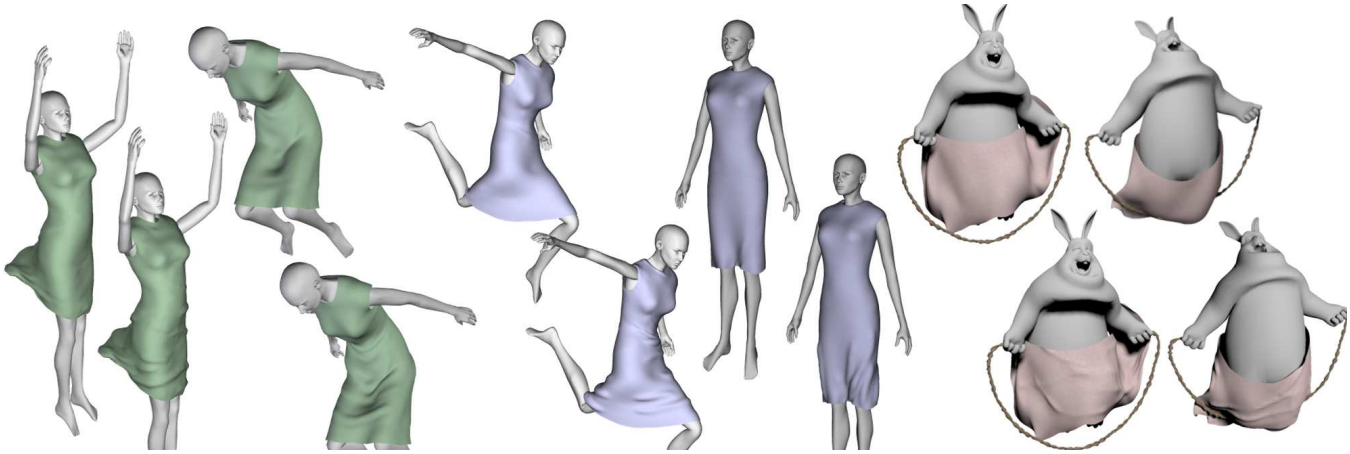


Figure 13: *Wrinkling garment simulations, before (top) and after (bottom).*

method, our solution does not currently handle extreme wrinkles that can appear when a piece of cloth is highly compressed. While those can be generated by using a positive offset to place wrinkle-deformers above the mesh, the projection method used to apply the deformation to the mesh vertices restricts us to height-field shapes.

Another geometric feature we did not address here is the sharp wrinkles that appear where cloth is put under tension. Here, we could use the stretch tensor to generate these wrinkles as well using lines of maximal elongation, given by the eigenvector of largest eigenvalue larger than one in the tensor.

Lastly, our method relies on the presence of a rest-shape, which is typically available in simulation settings. A promising future work would be to combine our method with some pre-process extracting such a rest-shape from a static mesh representing a worn garment or from a motion sequence, such as cloth motion capture.

References

BLOOMENTHAL, J., Ed. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics (SIGGRAPH) 21*, 3.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. *ACM Transaction on Graphics (SIGGRAPH) 21*, 3.

CHOI, K.-J., AND KO, H.-S., Eds. 2005. *Advanced Topics on Clothing Simulation and Animation. SIGGRAPH Courses*.

CUTLER, L., GERSHBEIN, R., WANG, X., CURTIS, C., MAIGRET, E., PRASSO, L., AND FARSON, P. 2005. An art-directed wrinkle system for CG character clothing. *Proc. of ACM/EG Symp. on Comp. Animation (SCA)*.

DE AGUIAR, E., SIGAL, L., TREUILLE, A., AND HODGINS, J. 2010. Stable spaces for real-time clothing. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4.

DECAUDIN, P., JULIUS, D., WITHER, J., BOISSIEUX, L., SHEFFER, A., AND CANI, M.-P. 2006. Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum (Eurographics) 25*, 3.

ENGLISH, E., AND BRIDSON, R. 2008. Animating developable surfaces using nonconforming elements. *ACM Transaction on Graphics (SIGGRAPH) 27*, 3.

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

KIMMERLE, S., WACKER, M., AND HOLZER, C. 2004. Multilayered wrinkle textures from strain. *VMV 225–232*.

MCCORMACK, J., AND SHERSTYUK, A. 2001. Creating and rendering convolution surfaces. *Computer Graphics Forum 17*, 113–120.

MULLER, M., AND CHENTANEZ, N. 2010. Wrinkle meshes. In *Proc. of ACM/EG Symp. on Comp. Animation (SCA)*.

PENNEC, X., FILLARD, P., AND AYACHE, N. 2006. A riemannian framework for tensor computing. *Int. J. of Comp. Vision 66*, 1.

POPA, T., ZHOU, Q., BRADLEY, D., KRAEVOY, V., FU, H., SHEFFER, A., AND HEIDRICH, W. 2009. Wrinkling captured garments using space-time data-driven deformation. *Computer Graphics Forum (Eurographics) 28*, 2.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture mapping progressive meshes. In *ACM SIGGRAPH'01*, 409–416.

SHEFFER, A., LEVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. ABF++: Fast and robust angle based flattening. *ACM Transaction on Graphics 24*, 2, 311–330.

SHEFFER, A., HORMANN, K., LEVY, B., DESBRUN, M., AND ZHOU, K. 2007. Mesh parameterization: Theory and practice. *ACM SIGGRAPPH, course notes*.

SHEWCHUK, J. R. 2002. Constrained delaunay triangulations. Software available at http://www.eecs.berkeley.edu/ jrs/.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. *SIGGRAPH'98*.

TALPAERT, Y. 2003. *Tensor Analysis and Continuum Mechanics*. Springer.

TANG, K., AND CHEN, M. 2009. Quasi-developable mesh surface interpolation via mesh deformation. *IEEE Transactions on Visualization and Computer Graphics 15*, 3, 518–528.

THOMASZEWSKI, B., PABST, S., AND STRASSER, W. 2009. Continuum-based strain limiting. *Computer Graphics Forum (Eurographics) 28*, 2.

WANG, H., HECHT, F., RAMANOORTHI, R., AND O'BRIEN, J. 2010. Example-based wrinkle synthesis for clothing animation. *ACM Transactions on Graphics (SIGGRAPH) 29*.