Animating Wrinkles by Example on Non-Skinned Cloth

Javier S. Zurdo, Juan P. Brito, and Miguel A. Otaduy

Abstract—The simulation of cloth with rich folds and wrinkles is a computationally expensive process. In this paper, we introduce an example-based algorithm for fast animation of plausible cloth wrinkles. Our algorithm does not depend on a character's pose, therefore it is valid for loose dresses, curtains, etc., not just cloth defined by skinning techniques. Central to our approach is a correspondence between low and high-resolution cloth deformations, both at the training and synthesis stages. Based on this correspondence, we define an algorithm for synthesizing cloth wrinkles as a function of the deformation of a low-resolution cloth and a set of example poses. We demonstrate the animation of plausible high-resolution wrinkles at high frame rates, suitable for interactive applications such as video games.

Index Terms—Cloth animation, data-driven methods, pose-space deformation

1 Introduction

Cloth is an important component of the expressiveness of an animated character and its motion. However, cloth is complex to simulate due to several reasons: it requires many degrees of freedom in order to capture folds and wrinkles in a realistic manner; its deformation is modeled using nonlinear energy functions; and avoidance of interpenetrations requires costly self-collision processing.

One common approach to accelerate computations of high-resolution models in computer graphics is to find a suitable low-resolution subspace that captures the important behavior, and define a mapping from the low-resolution subspace to the high-resolution domain. This approach has been followed for cloth-wrinkle animation too.

There are successful data-driven methods that animate high-resolution wrinkles on shirts and pants using a character's skeletal pose as low-resolution subspace [1], [2]. Unfortunately, wrinkles on generic cloth, such as a loose dress or a curtain, cannot be defined by a static mapping from a character's pose. Instead of using skeletal pose as subspace, several pieces of evidence show that, for skin, wrinkle formation can be expressed as a function in the reduced domain of local large-scale deformation [3], [4], [5]. And a similar observation has been recently exploited for procedural wrinkle formation on cloth [6], [7]. The conclusion we can draw is that even though real cloth wrinkles require a large number of degrees of freedom to capture their true diversity, they can be approximated in a plausible manner in the reduced domain of local large-scale deformation.

In this work, we propose a method for cloth animation that captures low-resolution cloth dynamics efficiently, and learns the mapping to high-resolution wrinkle deformations in a data-driven manner. Our work builds on three major components.

- A multiresolution representation of cloth animation, described in Section 3. A low-resolution representation captures cloth dynamics, while high-resolution wrinkles are defined in a quasistatic manner as a displacement from the low-resolution cloth.
- An example-based algorithm for synthesis of high-resolution cloth wrinkles, described in Section 4, that extends the approaches for wrinkle synthesis on skin.
- Methods for training and synthesis of wrinkle animation on practical examples, described in Section 5. One key aspect of our methods is that the training data sets are composed of high- and low-resolution cloth animations in close correspondence, generated using tracking constraints.

Our example-based algorithm for wrinkle synthesis is highly parallelizable. We have implemented it on graphics hardware, obtaining frame rates up to 125-250 fps on examples with meshes of 16K-25K triangles.

2 Related Work

Even though our work targets cloth animation, it does not follow traditional physically based approaches [8], [9], [10]. Such approaches, although highly versatile, are also computationally very expensive. Our discussion of related work focuses on procedural and example-based techniques for modeling deformable materials, in particular wrinkles in cloth or skin.

There are several procedural approaches for modeling wrinkles, on cloth [11], [12], skin [13], [14], [15], [16], or on both [17]. The methods of Wu et al. [3] and Rohmer et al. [7] are particularly relevant for our work. They formulate

- *J.S. Zurdo and M.A. Otaduy are with the Department of Computer Science, Universidad Rey Juan Carlos—ETSII, C/Tulipan s/n, 28933 Mostoles-Madrid, Spain. E-mail: {javier.zurdo, miguel.otaduy}@urjc.es.*
- *J.P. Brito is with the Universidad Rey Juan Carlos—ETSII, C/Tulipan s/n, 28933 Mostoles-Madrid, Spain and the Department of Computer Science, Universidad Politecnica de Madrid, Spain. E-mail: juanpedro.brito@upm.es.*

Manuscript received 3 Mar. 2011; revised 27 Oct. 2011; accepted 20 Feb. 2012; published online 7 Mar. 2012.
Recommended for acceptance by J. Keyser.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-03-0050.
Digital Object Identifier no. 10.1109/TVCG.2012.79.

wrinkle synthesis as a function of the strain or stress measured in an underlying low-resolution model. We build on the success of these approaches and select low-resolution strain as the subspace domain for our model, but instead we define wrinkle formation using examples in a data-driven manner, without parameter tuning. Müller and Chentanez [18] compute high-resolution wrinkles by solving a simplified static deformation model on top of a coarse cloth simulation. Their approach also exploits indirectly the local low-resolution cloth strain to govern the formation of wrinkles.

In skin modeling, example-based approaches have been successfully used, e.g., to correct skin deformation for modeling arms [19], hands [20], [21], or the body [22]. They typically combine a fast, skeletal subspace deformation (SSD) with a nonlinear pose-space deformation (PSD) [19] that interpolates correction vectors among example poses. PSD was extended to support weighted (i.e., per vertex) pose space deformation (WPSD) [21], [23], which largely reduces the number of required example poses. The EigenSkin method [20] performs corrections to SSD, but derives a reduced basis from a PCA of the input examples. Some recent methods [24], [25] learn example-based corrections on sparse points and assume that these corrections can be smoothly interpolated. Most methods focus on (quasi-)static skin deformation, but it is worth pointing out skeleton-driven skin dynamics [26].

Example-based methods have also been used for wrinkle animation on cloth. Extending skinning techniques, Kim and Vendrovsky [1] designed a pose-space deformation technique that uses a character's skeletal pose as subspace domain. This technique works well for tight cloth, but not for non-skinned loose cloth. A similar recent approach by Wang et al. [2] improves the performance of data-driven wrinkle animation for tight cloth by learning the influence of skeletal joints on wrinkle formation. Another recent method by de Aguiar et al. [27] learns cloth motion in the space of body motions and, interestingly, it also learns dynamics effects, but it does not target high-resolution wrinkling. Feng et al. [28] have designed a data-driven cloth animation approach that does not require a character's pose, and learns the mapping from low-resolution to high-resolution cloth. However, unlike ours, their technique does not maintain a close correspondence between low-resolution and high-resolution cloth in the training data set, which complicates the definition of the mapping and limits the overall quality under extrapolation. Concurrently to our work, Kavan et al. [29] have developed a method for adding wrinkle details to cloth simulations, and they also learn wrinkle data from simulations of low- and high-resolution cloth synchronized using tracking. Our approach differs from theirs in the tracking method and the wrinkle model. Their tracking approach defines hard constraints between the low- and high-resolution cloth, using mass-weighted harmonic test functions. Their wrinkle model is a linear upsampling operator, whose shape functions are learned from example simulations. Instead, we propose a nonlinear interpolation model that incurs in a slightly higher cost but is considerably more compact.

There are also example-based methods for skin or cloth wrinkle modeling that describe wrinkle details as a function of local low-resolution deformation [4], [5] or local stress maps [6]. Our work shares similarities with these approaches, but it differs in the formulation and interpolation of example weights. It also differs conceptually in the sense that it can be seamlessly integrated in a dynamic cloth simulator.

In our work, we obtain cloth wrinkle examples using high-resolution simulations, but examples could also be obtained from real cloth using vision-based capture methods. Some solutions include marker-based approaches that detect high-resolution folds [30], markerless capture of large-scale cloth deformation [31], or multiview stereo extraction of wrinkles together with space-time deformation [32]. Closing the loop with physically based simulation techniques, there are also methods for estimating cloth simulation parameters from video [33].

3 Multiresolution Cloth Representation

Our cloth representation builds on two major observations. First, the most salient dynamic effects of cloth can be captured at low resolution. And second, plausible high-resolution wrinkles can be defined in the reduced domain of low-resolution deformations. Following these observations, we simulate the dynamics of a low-resolution cloth, upsample it, and then add high-resolution wrinkle displacements to represent the full cloth.

This model presents some limitations, which should be mentioned upfront. Fine-scale wrinkle dynamics cannot be captured, as wrinkles are defined quasistatically. The model captures only a limited set out of all the possible wrinkles that a piece of cloth might present. However, thanks to the example-based nature of the wrinkles animated by our algorithm, they preserve natural characteristics such as length, width, and consistency over time. This is the key reason why the wrinkles appear plausible despite the limitations.

We define a piece of cloth in a multiresolution manner exploiting subdivision schemes in order to establish a correspondence between resolutions. Specifically, we use a two-resolution representation, where each resolution of the cloth is a two-manifold surface. We use triangulated surfaces, although our algorithm is not restricted to this type of discretization.

As shown in Fig. 1right, our cloth representation employs three different surface meshes, one at low resolution and two at high resolution.

1. *Feature mesh*. It is the low-resolution mesh. It defines the large-scale deformation of the cloth as well as all dynamics effects. Its deformation, together with a database of examples, fully defines wrinkle formation.
2. *Smooth mesh*. It is obtained by subdividing the feature mesh a user-defined number of times, and can be regarded as an upsampled version of the feature mesh. It constitutes the mesh on top of which wrinkles are synthesized.
3. *Detail mesh*. It is obtained by adding high-resolution displacements onto the smooth mesh. The smooth and detail meshes fully share the connectivity, thus trivially defining their correspondence.

Fig. 1. Example-based wrinkle animation. On the left, two example poses, showing the low-resolution feature mesh, the high-resolution detail mesh, and the high-resolution detail transformed to the rest pose. On the right, a synthesis example not in the training data set, showing the feature mesh, the subdivided smooth mesh, and the resulting detail mesh.

We have used Loop subdivision to obtain the smooth mesh from the feature mesh. For each vertex v_s of the smooth mesh and its corresponding vertex v_d of the detail mesh, we define the wrinkle displacement as the difference of their positions $\mathbf{x}_d - \mathbf{x}_s$, expressed in a local reference frame of the vertex of the smooth mesh. To define the local reference frame, we use the normal vector at the vertex and the edge vector to an arbitrary neighboring vertex. Since the smooth mesh is computed as a subdivided version of the feature mesh, local mesh distortions are smoothed, and the local reference frame is fairly insensitive to the particular choice of edge vector.

The local definition of wrinkle displacement is invariant under rigid transformations and allows the combination of wrinkle details from different configurations under a common reference. Given the local reference frame, we can transform the wrinkle displacements from the rest configuration to a deformed configuration and add wrinkle displacements to the smooth mesh in arbitrary deformed configurations. In the next section, we describe how these wrinkle displacements are computed.

4 Example-Based Wrinkles

Our algorithm for computing wrinkle displacements uses as input 1) the deformation of the feature mesh and 2) the displacements between the detail mesh and the smooth mesh for a small set of example poses. In order to model wrinkle displacements for an arbitrary deformation of the feature mesh, we adapt the WPSD scheme by Kurihara and Miyata [21]. In the rest of this section, we describe our low-resolution deformation descriptor that defines the interpolation domain for WPSD, the adaptation of WPSD to our setting, and further optimizations obtained by the computation of WPSD weights on a sparse set of vertices.

4.1 Low-Resolution Deformation Descriptor

WPSD combines pose data (in our case, wrinkle displacements) by defining per vertex pose weights through an interpolation process, and then computing a weighted average of pose data using those weights. WSPD requires the definition of a reduced domain in which per vertex pose weights are interpolated. Unlike Kurihara and Miyata, we cannot employ a skeleton's configuration as the reduced domain, as we intend to apply wrinkle animation on non-skinned cloth.

Instead, we adapt the approach of Bickel et al. [4], and use as reduced domain a low-resolution deformation descriptor of the cloth. This deformation descriptor is based on the edge lengths of the low-resolution mesh, i.e., the feature mesh, and it constitutes, in essence, a metric of low-resolution strain. There are alternative low-resolution deformation descriptors, such as the one used by Ma et al. [5], consisting of a 3D offset plus membrane strain. Even though our strain metric is mesh dependent, note that we learn a function that relates strain to wrinkle detail for a specific mesh, therefore it is fairly insensitive to the particular choice of strain metric.

Given a feature mesh with e edges, we define a *feature vector* $\mathbf{f} = [f_1, \ldots, f_e] \in \mathbb{R}^e$, where each component is the ratio

$$f_i = \frac{l_i}{l_{i0}}, \tag{1}$$

between the current length l_i and the rest length l_{i0} of an edge. Note that we slightly modify the definition of feature vector by Bickel et al., as we use actual edge lengths instead of the change of edge length. Given that the synthesis of deformations uses differences of feature vectors, both definitions produce the same results.

4.2 Weighted Pose-Space Deformation

Given a feature mesh with feature vector $\mathbf{f}$ and a set of P poses with feature vector values $\{\mathbf{f}_i\}$, we compute wrinkle displacements onto the smooth mesh using WPSD. WPSD defines the displacement of a vertex as a weighted sum of the displacements of that vertex in all input poses. In particular, for each vertex of the detail mesh, we define a vector of pose weights $\mathbf{w} = \{w_1, \ldots, w_P\} \in \mathbb{R}^P$. The pose weights are computed using radial-basis-function (RBF) interpolation

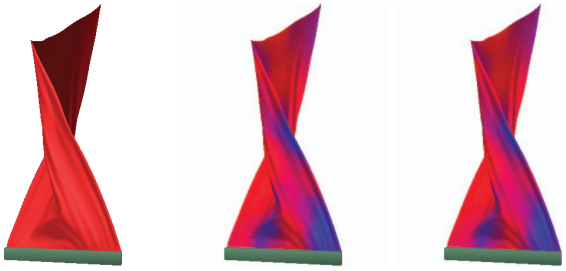

Fig. 2. Left: a certain deformation setting; Middle: densely evaluated weights for a certain pose; Right: sparsely evaluated weights, almost identical to the dense ones. Weights range from 0 (blue) to 1 (red).

$$\mathbf{w}(\mathbf{f}) = \sum \lambda_i \cdot \phi(\mathrm{dist}(\mathbf{f}, \mathbf{f}_i)). \tag{2}$$

The RBF weights $\lambda_i \in \mathbb{R}^P$ are precomputed such that pose weights fulfill the Kronecker delta for the database of poses, i.e., $w_i(\mathbf{f}_j)$ is 0 if $i \neq j$ and 1 if $i = j$, for poses i and j in the database.

We use RBFs with global support $\phi(r) = r$, as they avoid complex tuning of support radii for unevenly sampled data [34]. In WPSD, the feature distance $\mathrm{dist}(\mathbf{f}, \mathbf{f}_i)$ is computed as the euclidean distance weighted with a per vertex metric $\mathbf{M}$. This allows the use of a reduced database of poses, as data is interpolated with locally adapted weights. Then, and for our choice of RBF, the RBF for the ith pose can be computed as

$$\phi_i(\mathrm{dist}(\mathbf{f}, \mathbf{f}_i)) = (\mathbf{f} - \mathbf{f}_i)^T \mathbf{M} (\mathbf{f} - \mathbf{f}_i), \tag{3}$$

with $\mathbf{f}_i$ the feature vector of pose i in the database. We use the same weighting metric $\mathbf{M}$ as Bickel et al. [4], i.e., a diagonal matrix where each element m_{ii} is maximal on its corresponding edge of the feature mesh, and decays smoothly with distance. The weighting metric is precomputed in the rest configuration, and we use only the 16 largest edge-weights per vertex.

Given pose weights $\mathbf{w}$ for a certain vertex in the detail mesh, we normalize them in order to avoid extrapolation problems. Since we define wrinkle displacements in the rest configuration of the cloth, we obtain the wrinkle displacement for each vertex simply as a weighted sum of its displacement vectors in the database poses. Then, we compute the local reference system for the corresponding vertex in the smooth mesh, transform the wrinkle displacement, and add it to the smooth mesh to obtain the final position of the vertex in the detail mesh.

4.3 Sparse WPSD

We observed that, since the weighting metric $\mathbf{M}$ varies smoothly across the cloth surface, pose weights $\mathbf{w}$ vary smoothly as well. Therefore, we optimized our algorithm by computing WPSD weights only on a sparse set of vertices of the detail mesh.

In fact, given that the rest-state detail mesh is obtained through subdivision of the feature mesh, it seems like a natural choice to compute pose weights on the vertices of the detail mesh that are also original vertices of the feature mesh. Pose weights are interpolated to other vertices of the detail mesh using barycentric weights in the triangles of the feature mesh. Fig. 2 shows the clear similarity between pose weights interpolated from a sparse set and those computed independently on all vertices of the detail mesh.

Let us summarize the pipeline for data-driven modeling of wrinkles. Given a deformed feature mesh, we do the following:

1. Evaluate the component of the feature vector on each feature edge according to (1).
2. Compute the RBF value of each pose for each feature vertex, following WPSD as in (3). Recall that the weighting matrix $\mathbf{M}$ has up to 16 nonzero values in our implementation, hence this computation has an $O(1)$ cost per pose and per vertex.
3. Compute the pose weights for each feature vertex according to RBF interpolation as in (2).
4. Interpolate the pose weights to each detail vertex using barycentric interpolation.
5. Compute the wrinkle detail on each detail vertex through linear combination of pose details, using the readily available pose weights.
6. Subdivide the feature mesh using Loop subdivision to determine the smooth position of each detail vertex. In practice, we precompute the subdivision weights, and we implement subdivision as a fast linear combination.
7. Compute a local reference frame for each detail vertex, and add the wrinkle detail to the smooth position.

5 Training and Synthesis Procedures

In this section, we describe the overall procedures for obtaining a data set of example wrinkles (i.e., training), and for synthesizing wrinkled cloth animations at runtime. We discuss issues such as maintaining low-resolution and high-resolution correspondence, pose selection, contact handling, and parallel implementation.

5.1 Training

In the preprocess training procedure, we simulate both a low- and a high-resolution cloth. They will constitute the feature mesh and the detail mesh of the example poses. We also solve contact on both the low- and high-resolution cloth.

Without special treatment, the low-resolution and high-resolution simulations may diverge. But, in order to guarantee that displacement information is limited to fine-scale wrinkles (otherwise, the initial observations that inspire our model would not hold), it is important that the low-resolution cloth actually constitutes a low-resolution version of the high-resolution cloth during the complete training sequence.

Therefore, we have adopted a mechanism such that the high-resolution cloth *tracks* the motion of the low-resolution cloth. In fact, we use the TRACKS method by Bergou et al. [35]. We subdivide the low-resolution mesh at each animation frame, in the same way as the feature mesh is subdivided to obtain the smooth mesh, and use this subdivided mesh as *guide* mesh in the context of the TRACKS algorithm. Then, we set TRACKS constraints between the guide mesh and the high-resolution mesh, which acts as the *tracked* mesh.

Given a training animation sequence composed of N pairs of low-resolution and high-resolution meshes, we automatically select a small representative set with P pairs

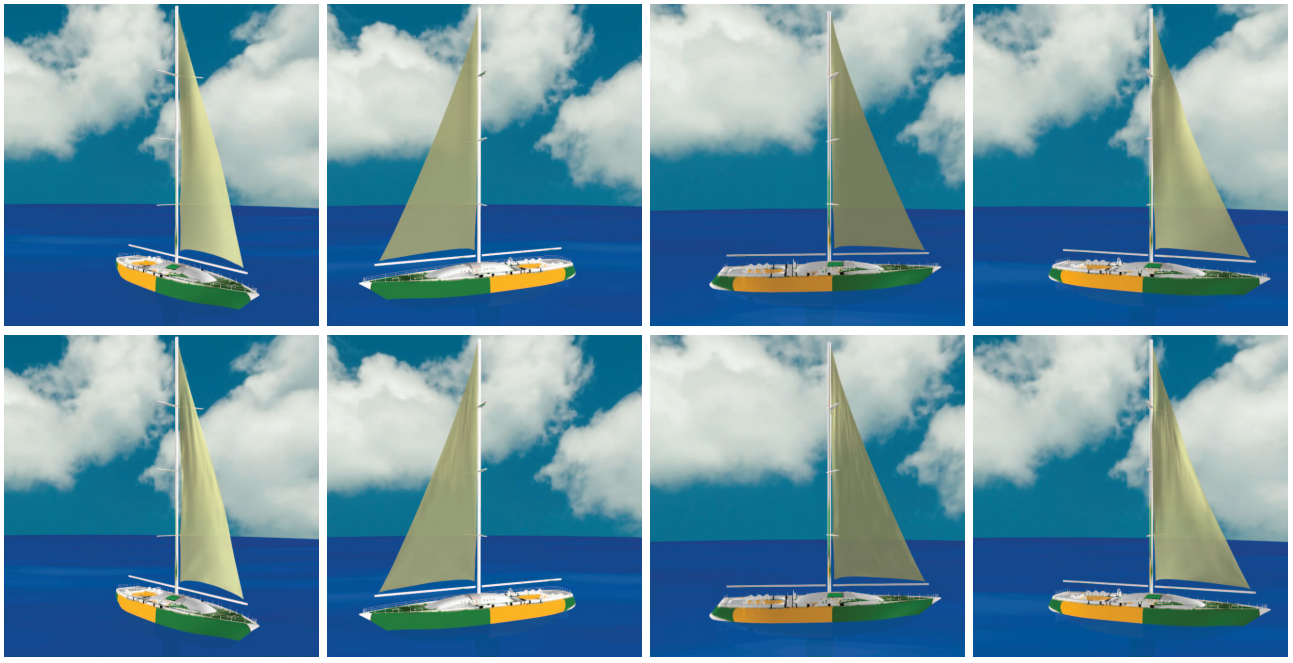

Fig. 3. Animation of wrinkles by example on a sail. The top row shows the low-resolution cloth simulation, and the bottom row shows the wrinkles animated by our algorithm. These results were obtained on a test sequence not in the training data set. In this particular example, we did not handle collisions and obtained a frame rate of 250 fps on a sail with 16K triangles, a $30\times$ speedup compared to a dynamics simulation (without collision handling) at the same resolution.

of meshes that capture well the spectrum of wrinkles across the complete training sequence. These P pairs of meshes constitute the example poses (both feature and detail meshes) for our interactive wrinkle animation algorithm. We initialize the P example poses with the rest pose (i.e., the rest feature and detail meshes), and then grow the database in a greedy manner. To add a new pose to the database, we resynthesize the complete training sequence using the current database, and pick the pose whose detail mesh suffers the largest L^2 error with respect to the input sequence. Section 6.1 reports error statistics as a function of the size of the database.

5.2 Synthesis

In the online animation synthesis procedure, we first simulate the low-resolution cloth defined by the feature mesh. We solve contact on the feature mesh with an extra safety distance to avoid interpenetrations on the detail mesh, and the resulting animation defines the overall deformation and all dynamics effects. Then, we subdivide the feature mesh to obtain the smooth mesh and we apply our example-based wrinkle computation to obtain the detail mesh.

Note that we do not solve contact on the detail mesh, only on the feature mesh. Then, there is no absolute guarantee that the detail mesh is collision free, but in most animations collisions do not occur or are hardly noticeable, thanks in part to the safety collision distance, but also because the wrinkle computation uses example detail meshes that are collision free.

The pipeline for wrinkle computation, outlined in Section 4.3, is highly parallelizable, and can be performed on massively parallel platforms such as GPUs. Specifically, after computing the low-resolution cloth simulation on the CPU, we compute, also on the CPU, pose weights on the feature vertices using WPSD. Then, we transfer to the GPU the vertex positions of the feature mesh and the per vertex pose weights. The rest of the computations are executed entirely on the GPU, including interpolation of the sparse weights, subdivision to obtain the smooth mesh, and the weighted average of wrinkle displacements to produce the final detail mesh. Most of the operations involve linear operators with just a few operands, which further increases the possibilities for high performance computation.

6 Results and Evaluation

We have tested our algorithm on four different non-skinned cloth examples: a twisting curtain (see Fig. 4), a boat sail (see Fig. 3), a flag on the wind (see Fig. 7), and a dancer's dress (see Fig. 9). The motion of the curtain's bar, the boat, and the dancer, could be described in a reduced domain and serve potentially as a low-dimensional domain for pose-space deformation. However, cloth motion cannot be skinned, i.e., it cannot be described as a function in the reduced domain, because of the effects of dynamics. With our method, on the other hand, we simulate the large-scale dynamics of the cloth, and we obtain an overall motion that does not appear skinned. Then, using our example-based wrinkle synthesis, the high-resolution animation shows plausible yet computationally efficient wrinkles.

All our experiments were tested on a 3.4 GHz AMD Phenom II x4 965 processor machine, with 4 GB of RAM, and a NVidia $GTX260$ graphics card. We have implemented the parallel wrinkle displacement computation on CUDA. For low-resolution cloth simulation, we have used mass-spring models with continuous collision detection and constraint-based collision response.

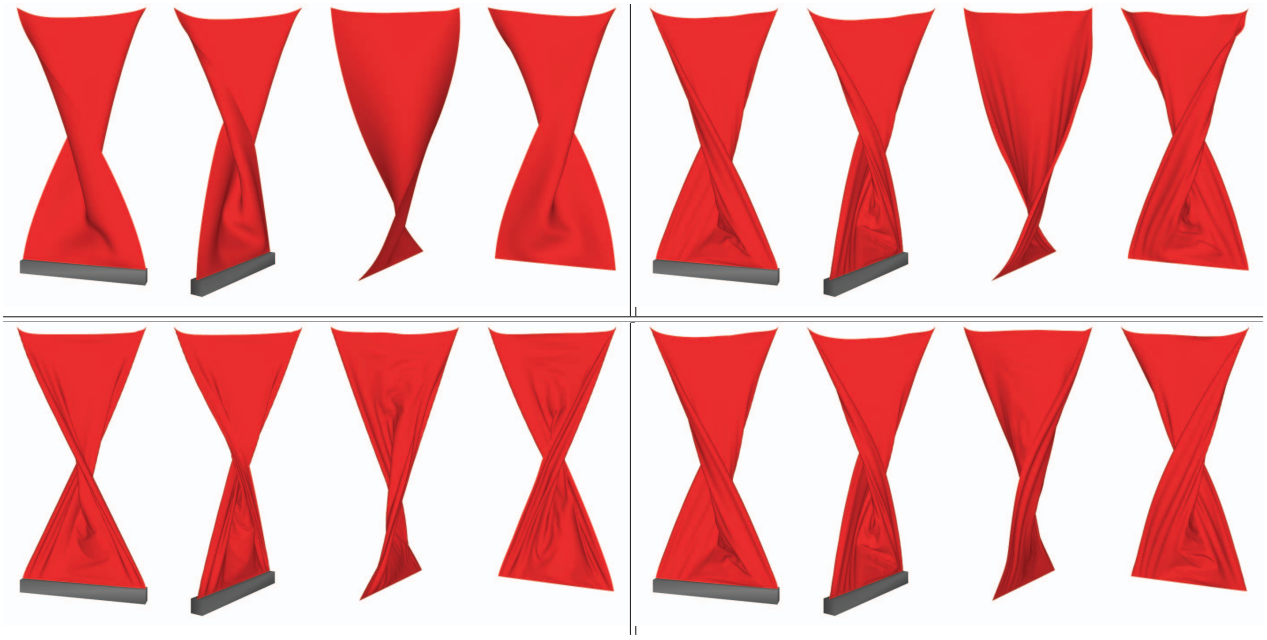

Fig. 4. Comparison of results obtained with four different animation methods. Top left: coarse simulation; Top right: our wrinkle animation method; Bottom left: high-resolution simulation; Bottom right: high-resolution simulation tracking the coarse simulation. In each method, the two leftmost images are obtained from the training sequence, while the two rightmost images are obtained from a different data set. Even though the overall dynamics differ from the full high-resolution simulation, our method produces plausible wrinkles under much higher frame rates (125 fps versus 2.5 fps).

6.1 Choice of Database

As indicated in Section 5.1, the selection of example poses for the database tries to minimize the fitting error in a training sequence in a greedy manner. In Fig. 5, we show a plot of the fitting error as a function of the size of the database for the curtain example. We found that a database with six example poses typically fit well the training sequence in our experiments, hence we used six example poses for all subsequent results, although other database sizes are also possible.

The accompanying video shows simulation results for training sequences as well as for novel sequences not used for training. These tests allow us to evaluate the ability to extrapolate wrinkle formation to different large-scale motions. For optimal results, the training sequence should cover the range of motions of the target simulation, otherwise the extrapolation of wrinkles will not produce plausible results. To stretch the applicability of our method, we have compared twist and shear motions for the curtain example, as shown in Fig. 6. We have generated separate

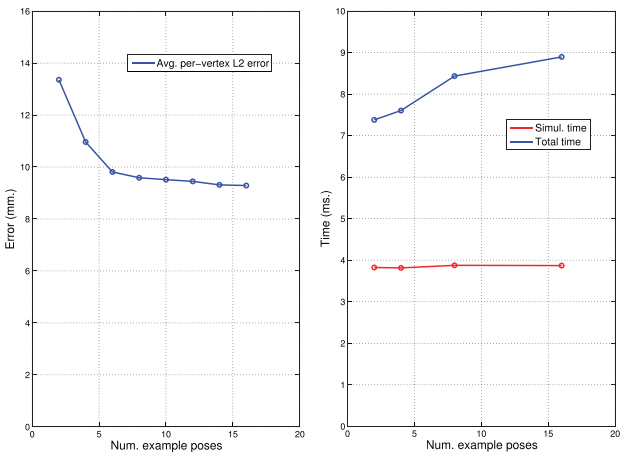

Fig. 5. Error and performance statistics for the curtain example as a function of the number of example poses in the database. Error is computed as per vertex L^2 error averaged over all vertices and all simulation frames. The performance plot shows the average per frame coarse-dynamics simulation cost, which is independent of the database, and the average total per frame cost including subdivision, wrinkle computation, and rendering (all three accelerated on the GPU).

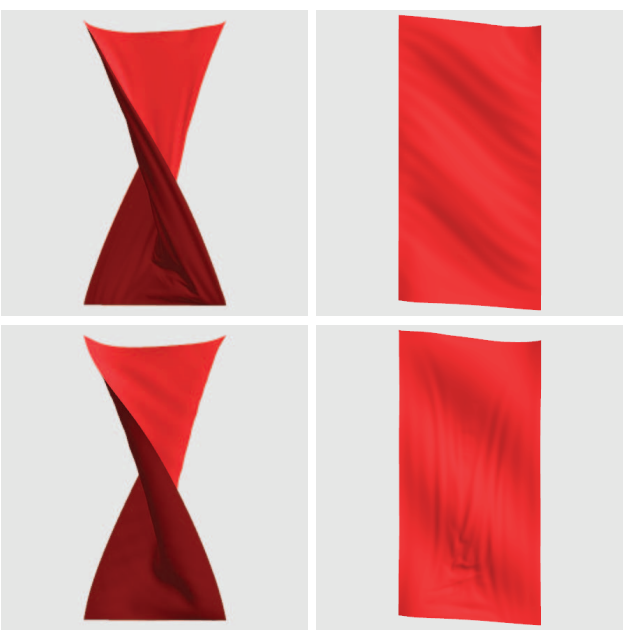

Fig. 6. Top row: twist and shear deformations using example poses from twist and shear training sequences, respectively. Bottom row: the example poses from the shear training sequence are applied to a twist deformation and vice versa. Our method fails to produce plausible wrinkles in such extreme situations.

TABLE 1
Complexity and Cost per Time Step (in Seconds) of Our Benchmarks

Benchmark	tris LR	tris HR	coarse	ours	coarse + subdiv.	full sim.	tracked
Curtain	400	25600	0.004	0.008	0.006	0.408	0.349
Sail	256	16384	0.001	0.004	0.003	0.117	0.125
Flag	400	6400	0.003	0.009	0.007	0.048	0.050
Dress	381	24384	0.071	0.075	0.073	0.746	0.901

*The second and third columns list the number of triangles of the low-resolution (feature) and high-resolution (detail) meshes in the three benchmarks. Then, we include the average cost per time step, in seconds, for: 1) a low-resolution simulation (*coarse*), 2) our method including low-resolution simulation and wrinkle synthesis (*ours*), 3) the low-resolution simulation plus rendering a subdivided mesh (*coarse + subdiv.*), 4) a full simulation at high resolution (*full sim.*), and 5) a simulation at high resolution that tracks the low-resolution simulation (*tracked*). In the curtain, sail, and flag examples, one rendered frame corresponds to one time step of simulation. In the dress example, however, the given timings correspond to a 3 ms time step in the case of our algorithm, and to a 1 ms time step in the case of the full simulation. Therefore, the total speedup in the dress example is even larger.*

databases using twist and shear training sequences, and we have applied the twist database to a shear simulation and vice versa. Such extreme situations induce failure cases for our method, as shown in the figure.

6.2 Performance

We compare, both in terms of performance and wrinkle quality, our results to three other simulation settings: 1) a coarse simulation subdivided to the same high-resolution that we use (this is equivalent to displaying the smooth mesh), 2) a full high-resolution simulation with contact handling on the high-resolution mesh, and 3) a tracked high-resolution simulation with contact handling on the high-resolution mesh as well (following the same procedure we use for training). The reason why we include the tracked example is that it shares the large-scale motion with our results, and helps us better evaluate wrinkle formation on its own. The full high-resolution simulation, instead, may present a large-scale motion that diverges over time.

In all examples, we simulate low-resolution cloth meshes with a few hundred triangles, and we animate high-resolution wrinkles on meshes with 6,400 to 25,600 triangles. As shown in Table 1, the performance of our algorithm is comparable to the subdivided coarse simulation, but we obtain high-resolution wrinkles which add a high degree of realism to the scenes. In the curtain example, the dynamics simulation runs at 250 fps, while the complete animation runs at 125 fps. The rest of the time is spent on wrinkle displacement computation and rendering. The high frame rate achieved makes our method suitable for video game applications, with a good balance of speed and quality. In the curtain and dress benchmarks, contact is handled on the low-resolution cloth, as discussed in Section 5.2. In the sail and flag examples, however, handling contact was not necessary, and that is the reason for higher frame rates. However, even in the sail scene, example-based wrinkle animation provides a $30\times$ speedup over a full dynamics simulation. In the curtain, sail, and flag examples, one rendered frame corresponds to one time step of simulation. In the dress example, however, due to time-step limitations, the given timings correspond to a 3 ms time step in the case of our algorithm, and to a 1 ms time step in the case of the full simulation. Then, our algorithm has a twofold advantage over a full simulation: faster processing per time step and larger time steps. In practice, we obtain a $30\times$ speedup over a full simulation.

In the curtain example, we have also evaluated performance as a function of the size of the database. As shown in Fig. 5, there is an offset cost due to the simulation of cloth dynamics on the low-resolution mesh. This task is dominated by contact handling to avoid self-collisions. Then, the rest of the cost is split between CPU-based coarse RBF weight computation, and GPU-based subdivision, weight interpolation, wrinkle detail combination, and rendering. The plot indicates that this cost is roughly linear in the size of the database, as it is dominated by wrinkle detail combination.

The precomputation stage of our method can take several minutes, depending on the size of the training sequence. This precomputation cost is dominated by contact handling on the tracked high-resolution simulation, therefore per frame timings correspond to those shown in Table 1 for the *tracked* column. The selection of the example poses in the database takes only a few seconds.

6.3 Discussion

The tests executed on sequences that were not in the training data set demonstrate a plausible extrapolation of wrinkle behavior. As a challenging example, the dress example shown in the video uses the first (slow) part of the dance as training sequence, and then plausible wrinkles are animated on the second (faster) part of the dance. As mentioned earlier in Section 3, one key reason why animated wrinkles appear plausible is that they preserve natural characteristics such as length, width, and consistency over time, which is a feature of the adopted pose-space interpolation method. Careful examination shows that our animated wrinkles appear more repetitive than actual simulated ones, but this is an expected limitation due to the small number of examples used during synthesis. In the flag example (see Fig. 7), wrinkling behavior is dominated by dynamics. Our technique, on the other hand, models wrinkles as a quasistatic phenomenon; therefore, it cannot capture the richness of wrinkles in a flag, with effects such as travelling waves.

It is not possible to evaluate wrinkle accuracy using traditional L^2 error metrics, as qualitatively similar deformations may contain wrinkles at slightly different positions, inducing large per vertex errors. Instead, we have evaluated the curvature content of our animated wrinkles over time, comparing it to that in a coarse (subdivided) simulation, a full high-resolution simulation, and a tracked high-resolution

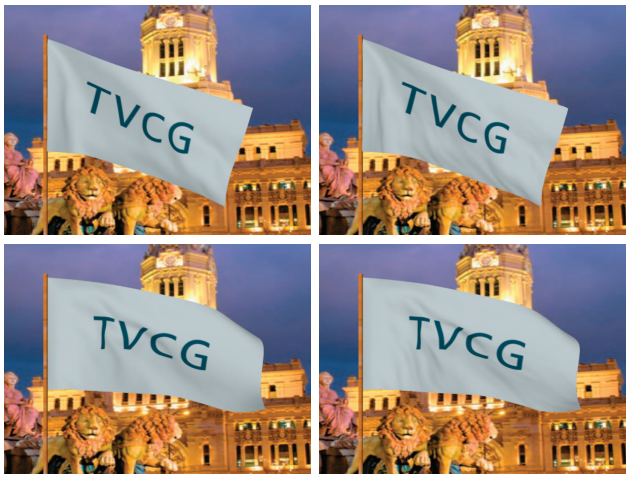

Fig. 7. Left column: two frames of a (subdivided) low-resolution flag simulation. Right column: the same simulation with wrinkles added by our algorithm.

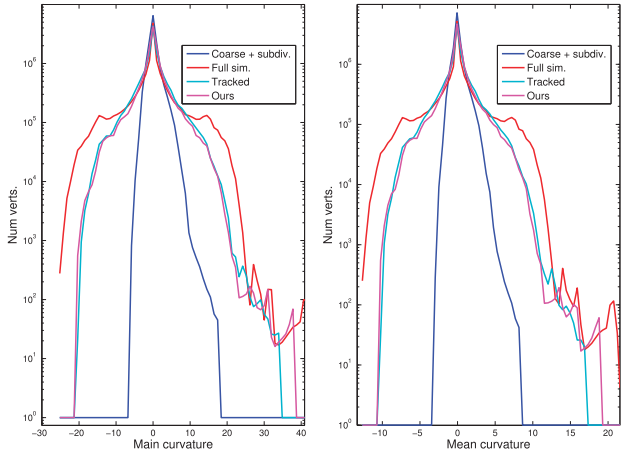

Fig. 8. Histograms of main curvature (left) and mean curvature (right) for the curtain demo, under different simulation settings.

simulation. In particular, Fig. 8 shows histograms of main and mean curvature for the curtain example (computed using the method in [36]). The curvature content with our method

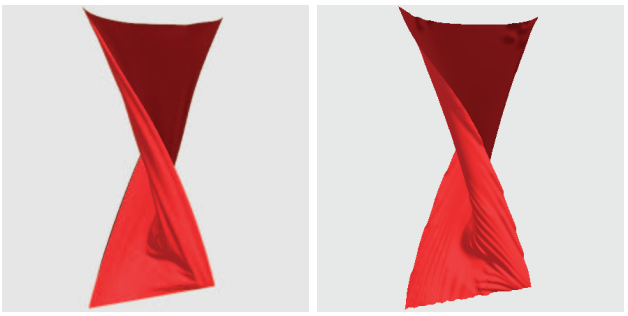

Fig. 10. Our technique (left) produces qualitatively similar results to procedural wrinkle synthesis methods (right: result generated with the method of Rohmer et al. [7]) at a lower computational cost.

is very close to that in the tracked simulation, and approximates that in the full high-resolution simulation, although the finest wrinkles and folds are missed due the use of coarse dynamics.

Some of the results produced in our examples could be achieved perhaps using other techniques. In situations with shallow wrinkles, such as the sail example in Fig. 3, one could consider using bump mapping to render wrinkles. Then, our method could be modified to interpolate bump data in pose-space. Our technique can be regarded as a variant of displacement mapping, with dynamically computed displacements. To demonstrate the impact of dynamic wrinkles on the quality of the animations, the accompanying video compares the dress example using our technique and using a fixed displacement map (picked from one of the example poses). Yet another possible alternative to our method would be to use procedural wrinkle synthesis techniques, such as the one by Rohmer et al. [7], which are more artist friendly. As shown in Fig. 10, with the appropriate settings, Rohmer's technique produces results very similar to ours, but at a notably higher computational cost.

There are additional advantages of our wrinkle animation algorithm. One is that, with our combination of low-resolution cloth simulation plus high-resolution example-based wrinkles, our contact solver had no problems with

Fig. 9. Example-based wrinkle animation of a dancer's dress. Top row: a low-resolution simulation with only 381 triangles defines the dynamics, the large-scale deformation, and response to contact. Bottom row: high-resolution dress with wrinkles animated efficiently from six example poses and the deformation of the low-resolution cloth.

constraint satisfaction or pinching. High-resolution interpenetrations might appear, but for video game applications this is arguably a lesser problem than pinching or a high number of solver iterations. For the full high-resolution simulation of the curtain and the dress, on the other hand, we found it impossible to ensure a reasonably controllable computational cost without pinching or robustness problems. Interestingly, for the curtain example, the cost of the tracked simulation is smaller than the cost of the full simulation, because the contact configurations are simpler, even though it requires solving two simulations in one. Finally, the application of our algorithm to the training data sets also shows another possible applicability to video games, i.e., the efficient compression and decompression of prerecorded sequences.

7 Limitations and Future Work

As discussed throughout the paper, one clear limitation of our work stems from the very nature of example-based techniques: it is not possible to capture the full spectrum of cloth wrinkles in a reduced domain. Another clear limitation stems from the use of low-resolution dynamics, which cannot capture the full dynamic spectrum of a true high-resolution cloth. This issue also affects the creation of the example poses during training, where the high-resolution cloth tracks a low-resolution cloth and does not exhibit full high-resolution dynamics. Due to these two limitations, our technique is probably not suited for feature film production. However, it offers an excellent balance between performance and wrinkle quality for interactive applications such as video games. The computational cost grows approximately linearly with the number of poses; therefore, a small number of poses (e.g., below 10) must be used to achieve high performance.

Another limitation is the possibility to incur in cloth interpenetration, as we resolve contact on the low-resolution cloth. Interestingly, handling contact on the high-resolution cloth during training implies that much of the collision response behavior is inherently present in the examples. Moreover, as mentioned in the previous section, contact handling on the low-resolution cloth has other benefits for interactive applications such as a more controllable cost.

Currently, our method handles only quasistatic wrinkle formation, and it would be interesting to enhance the reduced domain deformation descriptor with dynamics information, in order to model effects of wrinkle dynamics. Due to the use of subdivision schemes, our cloth models are limited in terms of the features they can present in the undeformed state. We estimate that this limitation can be overcome by substituting our smooth mesh with a representation similar to multiresolution meshes [37].

We also plan to explore transfer of wrinkles from one piece of cloth to a different one, and it would be interesting to evaluate the suitability of our method from a perceptual point of view.

Acknowledgments

This project has been supported in part by the Spanish Ministry of Science and Innovation (project TIN2009-07942). The character in the dance scene is courtesy of Disney Animation. The authors would also like to thank Sara Schvartzman and Jorge Gascón for help with demos, Ignacio García for demo ideas, and the GMRV group at URJC Madrid.

References

[1] T.-Y. Kim and E. Vendrovsky, "Drivenshape - A Data-Driven Approach to Shape Deformation," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* 2008.

[2] H. Wang, F. Hecht, R. Ramamoorthi, and J. O'Brien, "Example-Based Wrinkle Synthesis for Clothing Animation," *ACM Trans. Graphics,* vol. 29, no. 4, pp. 107:1-107:8, July 2010.

[3] Y. Wu, P. Kalra, and N. Magnenat-Thalmann, "Simulation of Static and Dynamic Wrinkles of Skin," *Proc. Computer Animation,* pp. 90-97, 1996.

[4] B. Bickel, M. Lang, M. Botsch, M.A. Otaduy, and M. Gross, "Pose-Space Animation and Transfer of Facial Details," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* pp. 57-66, 2008.

[5] W.-C. Ma, A. Jones, J.-Y. Chiang, T. Hawkins, S. Frederiksen, P. Peers, M. Vukovic, M. Ouhyoung, and P. Debevec, "Facial Performance Synthesis Using Deformation-Driven Polynomial Displacement Maps," *ACM Trans. Graphics (Proc. ACM SIGGRAPH Asia),* vol. 27, no. 5, article 121, 2008.

[6] L.D. Cutler, R. Gershbein, X.C. Wang, C. Curtis, E. Maigret, L. Prasso, and P. Farson, "An Art-Directed Wrinkle System for CG Character Clothing and Skin," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* 2005.

[7] D. Rohmer, T. Popa, M.-P. Cani, S. Hahmann, and A. Sheffer, "Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles," *ACM Trans. Graphics,* vol. 29, no. 5, pp. 157:1-157:8, 2010.

[8] D.H. House and D.E. Breen, *Cloth Modeling and Animation.* AK Peters, 2000.

[9] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of Clothing with Folds and Wrinkles," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* 2003.

[10] K.-J. Choi and H.-S. Ko, "Advanced Topics on Clothing Simulation and Animation," *Proc. ACM SIGGRAPH Courses,* 2005.

[11] S. Hadap, E. Bangarter, P. Volino, and N. Magnenat-Thalmann, "Animating Wrinkles on Clothes," *Proc. IEEE Visualization Conf.,* 1999.

[12] P. Decaudin, D. Julius, J. Wither, L. Boissieux, A. Sheffer, and M.-P. Cani, "Virtual Garments: A Fully Geometric Approach for Clothing Design," *Computer Graphics Forum (Proc. Eurographics),* vol. 25, no. 3, pp. 581-590, 2005.

[13] N. Magnenat-Thalmann, P. Kalra, J.L. Lévque, R. Bazin, D. Batisse, and B. Queleux, "A Computational Skin Model: Fold and Wrinkle Formation," *IEEE Trans. on Information Technology in Biomedicine,* vol. 6, no. 4, pp. 317-323, Dec. 2002.

[14] Y. Bando, T. Kuratate, and T. Nishita, "A Simple Method for Modelng Wrinkles on Human Skin," *Proc. Pacific Graphics,* 2002.

[15] Y. Zhang and T. Sim, "Realistic and Efficient Wrinkle Simulation Using an Anatomy-Based Face Model with Adaptive Refinement," *Proc. Computer Graphics Int'l,* pp. 3-10, June 2005.

[16] K. Venkataraman, S. Lodha, and R. Raghavan, "A Kinematic-Variational Model for Animating Skin with Wrinkles," *Computers and Graphics,* vol. 29, no. 5, pp. 756-770, 2005.

[17] C. Larboulette and M.-P. Cani, "Real-Time Dynamic Wrinkles," *Proc. Computer Graphics Int'l,* 2004.

[18] M. Müller and N. Chentanez, "Wrinkle Meshes," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* 2010.

[19] J.P. Lewis, M. Cordner, and N. Fong, "Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation," *Proc. ACM SIGGRAPH,* pp. 165-172, 2000.

[20] P. Kry, D.L. James, and D.K. Pai, "EigenSkin: Real-Time Large Deformation Character Skinning in Hardware," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* pp. 153-159, 2002.

[21] T. Kurihara and N. Miyata, "Modeling Deformable Human Hands from Medical Images," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* pp. 357-366, 2004.

[22] S.I. Park and J.K. Hodgins, "Capturing and Animating Skin Deformation in Human Motion," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 25, no. 3, pp. 881-889, 2006.

[23] T. Rhee, J.P. Lewis, and U. Neumann, "Real-Time Weighted Pose-Space Deformation on the GPU," *Computer Graphics Forum (Proc. Eurographics),* vol. 25, no. 3, pp. 439-448, 2006.
[24] R.Y. Wang, K. Pulli, and J. Popović, "Real-Time Enveloping with Rotational Regression," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 26, no. 3, article 73, 2007.
[25] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman, "Context-Aware Skeletal Shape Deformation," *Computer Graphics Forum (Proc. Eurographics),* vol. 26, no. 3, pp. 265-274, 2007.
[26] S.I. Park and J.K. Hodgins, "Data-Driven Modeling of Skin and Muscle Deformation," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 27, no. 3, article 96, 2008.
[27] E. de Aguiar, L. Sigal, A. Treuille, and J.K. Hodgins, "Stable Spaces for Real-Time Clothing," *ACM Trans. Graphics,* vol. 29, no. 4, pp. 106:1-106:9, July 2010.
[28] W.-W. Feng, Y. Yu, and B.-U. Kim, "A Deformation Transformer for Real-Time Cloth Animation," *ACM Trans. Graphics,* vol. 29, no. 4, pp. 108:1-108:9, July 2010.
[29] L. Kavan, D. Gerszewski, A.W. Bargteil, and P.-P. Sloan, "Physics-Inspired Upsampling for Cloth Simulation in Games," *Proc. ACM SIGGRAPH,* 2011.
[30] R. White, K. Crane, and D. Forsyth, "Capturing and Animating Occluded Cloth," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 26, no. 3, article 34, 2007.
[31] D. Bradley, T. Popa, A. Sheffer, W. Heidrich, and T. Boubekeur, "Markerless Garment Capture," *ACM Trans. Graphics (Proc. ACM SIGGRAPH),* vol. 27, no. 3, p. 99, 2008.
[32] T. Popa, Q. Zhou, D. Bradley, V. Kraevoy, H. Fu, A. Sheffer, and W. Heidrich, "Wrinkling Captured Garments Using Space-Time Data-Driven Deformation," *Computer Graphics Forum (Proc. Eurographics),* vol. 28, no. 2, pp. 427-435, 2009.
[33] K.S. Bhat, C.D. Twigg, J.K. Hodgins, P.K. Khosla, Z. Popović, and S.M. Seitz, "Estimating Cloth Simulation Parameters from Video," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* 2003.
[34] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," *Proc. ACM SIGGRAPH,* pp. 67-76, 2001.
[35] M. Bergou, S. Mathur, M. Wardetzky, and E. Grinspun, "TRACKS: Toward Directable Thin Shells," *Proc. ACM SIGGRAPH,* 2007.
[36] S. Rusinkiewicz, "Estimating Curvatures and Their Derivatives on Triangle Meshes," *Proc. Symp. 3D Data Processing, Visualization, and Transmission,* 2004.
[37] D. Zorin, P. Schröder, and W. Sweldens, "Interactive Multiresolution Mesh Editing," *Proc. SIGGRAPH '97,* pp. 259-268, Aug. 1997.

Javier S. Zurdo received the degree in computer science from Universidad Rey Juan Carlos (URJC Madrid) in 2004. Between 2000 and 2004, he did additional training on information technology infrastructure, development, and operations. From 2003 to 2006, he worked for the "information and technologies services" at URJC, and from 2006 to 2011, he has worked as teaching and research assistant at URJC, within the GMRV research group. Currently, he is working as a director of computing services for the Leganés city council. His research interests include physically based animation, cloth animation, topology changes, directable animation, and virtualization systems.

Juan P. Brito received the degree in computer science from Universidad de Las Palmas de Gran Canaria, and the master's degree in graphics, games and virtual reality from Universidad Rey Juan Carlos (URJC Madrid) in 2010. Currently, he is pursuing the PhD degree with Universidad Politécnica de Madrid, under the Cajal Blue Brain project, and working as a research assistant with the GMRV group at URJC Madrid. His main interests include computational geometry, parallel algorithms, and high performance computing.

Miguel A. Otaduy received the BS degree in electrical engineering from Mondragón University, the MS and PhD degrees in computer science from the University of North Carolina at Chapel Hill, in 2000, 2003, and 2004, respectively. Currently, he is working as an associate professor in the Department of Computer Science's Modeling and Virtual Reality Group (GMRV) at Universidad Rey Juan Carlos (URJC Madrid). From 2005 to 2008, he was a research associate at ETH Zurich, and then he joined URJC Madrid. His main research interests include physically based computer animation, haptic rendering, collision detection, virtual reality, and geometric algorithms. He has published more than 50 papers in computer graphics and haptics, and has recently cochaired the program committees for the ACM SIGGRAPH/Eurographics Symposium on Computer Animation and the Spanish Computer Graphics Conference, both in 2010.