Volume 24 (2005), number 2 pp. 173–183

A Data-Driven Approach for Real-Time Clothes Simulation[†]

Frederic Cordier[1] and Nadia Magnenat-Thalmann[2]

[1] Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon 305-701, Korea
[2] MIRALab, University of Geneva, 24 rue du General Dufour, CH-1211 Geneva, Switzerland

Abstract

A data-driven approach for the real-time processing of clothes, particularly suitable for simulating dresses worn by virtual characters, is proposed. It starts, prior to real-time simulation, by analyzing cloth behavior in relation to the underlying skeleton movement from a presimulated sequence of the cloth obtained using any high-quality off-line simulators. The idea is to use this analysis to find an optimal combination of physics-based simulation and geometric approximation of the simulator; potentially colliding regions are defined on the cloth such that they will hold true for the skeleton movement that closely matches that of presimulated sequence. At runtime, using these analyses, our simulation process provides both visually pleasing results and performance, as long as the motion of the character remains sufficiently close to the original sequence used for the precomputation.

The key contributions of this paper are (1) efficient collision handling that prunes out potentially colliding objects by using the off-line simulation sequence as examples; (2) data-driven fix-up process for the coarse mesh simulation that deduces the gross behavior of the cloth; and (3) geometric approximation of the fine mesh deformation, responsible for details in the shape of the cloth such as wrinkles.

Keywords: cloth simulation, collision detection, cloth wrinkles, data–driven approach, geometric deformation, linear interpolation
ACM CCS: I.3.5 Computer Graphics: *Physically based modeling*

1. Introduction

Cloth plays a dominant role in almost every process important to humanoid digital creatures. The problem of simulating the behavior of clothes is one subject the graphics community has been grappling with since almost two decades ago.[1,2,3]

Relatively, little emphasis has been placed on the separate problem of how to achieve real-time performance in simulating the cloth. A number of strategies have been suggested, such as using simplifying assumptions for the physics model and/or collision detection [4,5]. A recent work by James and Fatahalian [6] suggests a different approach by adopting a data-driven method. These techniques do not suffice; however, when simulating fully dressed virtual characters in real-time, leaving the topic unexplored.

We present a data-driven method for simulating clothes worn by 3D characters in real-time. To effectively optimize the physics-based deformation, which is the bottleneck of the simulation, we use a coarse representation of the cloth mesh to drive the gross behavior in simulation. We consider that the gross cloth behavior is driven mainly by two separable contributions: the skeleton-driven movement of the character and the mechanical properties of the cloth. This consideration was partly inspired by the hybrid real-time simulation method proposed in Cordier and Magnenat-Talmann [7], where a hybrid deformation method is used to combine dynamic surfaces with skeleton-driven deformation (SDD). Unlike that method, however, our method exhibits significantly more efficient and realistic behavior. This effect is achieved by focusing on the analysis of cloth movements in relation to its associated skin surface, and adopting a learning strategy. The idea is to use the analysis of the presimulated sequence to identify the region largely explained by joint movement and to replace the physics-based simulation with geometric methods wherever possible.

[†]This work was done when at MIRALab.

Accepted April 2005

In our approach, the key ingredients of the new technique are associated with different facets of cloth simulation: first, our novel collision detection prunes out unnecessary collision tests by tightly localizing potentially colliding regions through the analysis of the cloth movement in relation to the skeleton. Second, we use the presimulated sequence to approximate the dynamic behavior of the coarse mesh geometrically wherever possible. Finally, fine details such as wrinkles are also simulated in a data-driven manner, by using the presimulated cloth sequence as examples. Subsequently, real-time animation of fully dressed human could be generated, which would be suitable for applications such as games where visual plausibility is more important than accuracy.

The remainder of this paper is organized as follows. We begin by reviewing previous approaches in Section 2. Section 3 gives an overview, followed by the description of our method for SDD in Section 4. The next two sections are dedicated to the simulation of gross behavior and the generation of wrinkles, respectively. After demonstrating results and performance in Section 7, we conclude with discussion and future work in Section 8.

2. Previous Work

The history of research on real-time cloth is relatively recent. Researchers have concentrated mainly on two aspects of real-time cloth animation: simulating the physical properties of garments and collision handling.

2.1. Numerical Solvers

Probably the most common technique for simulating the physical properties of clothes is the particle system. Simulation process is broken down into calculating the internal forces and solving the system of partial differential equations (PDE). The latter point has attracted much interest in the field of real-time applications, since it requires high computation power.

The explicit Euler method [8] has been one of the first numerical solvers. Unfortunately, this method is notorious for its instability when using large time steps and stiff equations. Several improvements have been proposed to reduce instability, such as the Verlet integration [9] and the explicit Euler combined with inverse dynamics [10,11]. Unfortunately, the simulation quality is sacrificed in favor of computation speed due to the approximations employed in these models.

The implicit Euler method presented by Baraff and Witkin [8] performs the computation not by using the derivative at the current time, but the predicted derivative at the next time step. Unlike explicit Euler integration, the implicit Euler method offers higher stability while using large time steps and clothes with stiff mechanical properties. A major drawback of this numerical solver, however, is the computation of a large linear system.

More recently, researchers worked on saving the computation time of the linear system solver. Desbrun *et al.* [6] proposed solving the linear system with a precomputed inverse matrix. Kang and Cho [5] proposed further optimization with a direct update formula for the positions and velocities of the cloth vertices. As indicated by the authors, these methods are not intended to provide a physically correct cloth animation. Our approach to that problem is a data-driven mass-spring system: the simulation is corrected with a set of functions built from the presimulated animation. By doing so, we bring the deformation of the mass-spring system closer to the original cloth behavior.

Another approach to fast garment deformations is the hybrid approach. They aim for a neat combination of physically based deformation and geometric deformation. Cordier and Magnenat-Talmann [6] proposed to segment the cloth into pieces and simulate these by different algorithms, depending on how they lie on the body surface and whether they adhere to it or flow over it. Others have noted that wrinkle deformation is geometric in nature and therefore can be computed with a geometric method. Wrinkles can be generated either by tessellating the cloth mesh [5] or rendering details on texture using bump mapping [12]. The main difficulty is defining a fold function that can simulate all kinds of wrinkle patterns. Moreover, determining the location and shape of wrinkles is left to artists. One of our contributions is a geometric wrinkling method that is ‘trained’ by using a presimulated cloth sequence rather than relying on users.

2.2. Collision Handling

Collision detection is usually one of the bottlenecks in real-time animation. The problem is particularly acute in the case of clothes because these objects are highly deformable. Several algorithms have been proposed to process robustly collisions in cloth simulation [3,13] without reaching real-time performance. Some other methods exploit graphics hardware to compute collisions on bump maps [11]; others use voxel trees that partition the space hierarchically [14]. Using frame coherency to reduce computation cost has been explored by Zhang *et al.* [15]. In this work, we propose a data-driven collision detection method; we use the presimulated sequence to localize the collision checks to neighboring cloth regions that have high probability to collide.

2.3. Data-Driven Approaches

The idea of building an interpolator from examples or presimulated data has proven to be a valuable tool in a variety of areas of CG, for example, for modeling a variety of human body shapes and for motion synthesis. The basic idea is to build an interpolation space filled with a set of pairs of input

Table 1: *The chosen strategies to save computation time.*

Simulation stages	Strategies to reduce the computation time
Simulating the gross movements on the coarse mesh	- Simplified mass-spring system - Implicit Euler integrator - Post-correction to maintain the cloth behaviour closer to the pre-simulated one
Collision detection on the coarse mesh	- Each vertex is enclosed into a collision hull that is rigidly attached to the skeleton - Extra collision detection is also computed between the floating regions and the skeleton joints
Modelling the cloth details on the fine mesh	- Geometric deformation with a linear function which coefficients are defined by linear regression on the pre-simulated animation

parameters and the targeted graphical objects. Cloth animation depends on a high number of parameters and therefore a data-driven approach is difficult to adapt. Very recently, James and Fatahalian [6] resented such an approach, where physics-based deformation and collision detection are both handled in a unified framework. By blending of precomputed orbits rather than using a mass-spring system, previous unseen results could be achieved, such as garments with stiff mechanical properties in real-time. However, they show little degrees of freedom (DoF) to the clothes under simulation; Instead of resorting to a data-driven approach for the entire simulation, we seek a neat combination of a data-driven approach with the mass-spring system. Unlike previous works, our simulator allows a much higher degree of interaction, as it is often needed in animating clothes on moving characters.

3. Overview of Our Approach

The primary focus of this paper is the development of a fast cloth simulator for real-time applications. Dynamic simulation of complex deformable models, however, can easily involve thousands of DoF. For example, a physics-based simulator would require several minutes to compute one frame of a cloth model worn by a character. Simulating large models directly would therefore be computationally impractical. In what follows, we present the chosen optimization strategies as well as the workflow of the method.

3.1. Optimization Strategies

Table 1 summarizes our strategies for optimizing the cloth simulation. Our simulator is based on two levels of deformation: the first deduces the gross cloth behavior by working on a coarse mesh with a physics-based approach whereas the second generates wrinkles on a fine mesh with a geometric method. The coarse mesh is generated by simplifying the original cloth mesh through segmentation. The reason for this choice is to lower the computation time; geometric

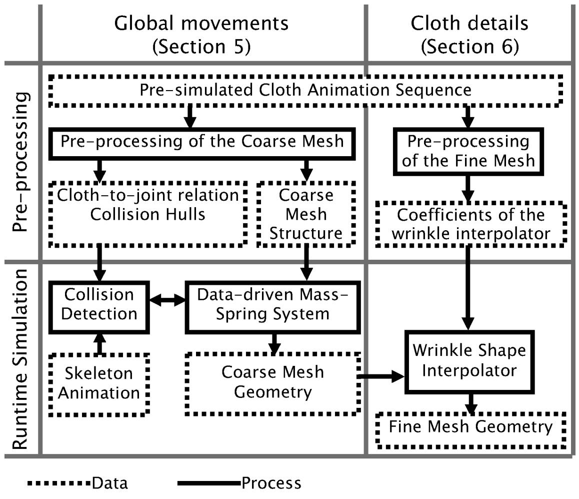
Global movements (Section 5)
Cloth details (Section 6)
Pre-processing
Runtime Simulation
Pre-simulated Cloth Animation Sequence
Pre-processing of the Coarse Mesh
Pre-processing of the Fine Mesh
Cloth-to-joint relation Collision Hulls
Coarse Mesh Structure
Coefficients of the wrinkle interpolator
Collision Detection
Data-driven Mass-Spring System
Skeleton Animation
Coarse Mesh Geometry
Wrinkle Shape Interpolator
Fine Mesh Geometry
Data
Process

Figure 1: *The workflow of our approach.*

methods are in general much faster than physically based ones [12].

Next, our simulator assumes that the clothes are worn on 3D characters, and that the character movement is the only external force acting upon the cloth. When observing the behavior of garment worn by a character, there are considerable correlations between the body motion and the movement of the garment. These correlations are especially clear for some clothes such as tight shirts and trousers. In our method we take advantage of these relationships to reduce the computation load on the mass-spring system and collision detection. We first construct the cloth-to-joint relation by analyzing a presimulated sequence of the cloth to be animated. We then reduce the number of vertices to be physically simulated by identifying the garment regions in which the shape follows that of the underlying skin. The cloth-to-joint relation enables us also to optimize collision detection by restricting the collision check to a small area around each vertex of the coarse mesh. Finally, we use the cloth shape of a presimulated cloth sequence to correct the physics-based simulation of the coarse mesh in order to match the original cloth behavior more closely.

3.2. Workflow

An overview of the workflow is given in Figure 1. The preprocessing stage involves generating the coarse mesh, computing the cloth-to-joint relation, and constructing the collision hulls and the interpolation functions for data-driven coarse mesh deformation and wrinkle animation.

The runtime simulator includes deformation of the coarse mesh using the simplified mass-spring system; a post-correction on the position and velocity of the mass points

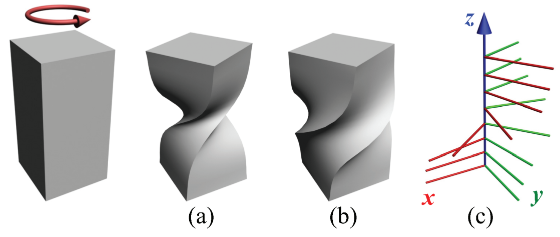

Figure 2: *Twisting of $\frac{2}{3}\pi$ radians: (a) the classical SDD, (b) the modified SDD in our approach, and (c) its corresponding coordinate systems.*

is processed in order to approach the presimulated cloth behavior. Collisions are handled by collision hulls the position of which is computed by our SDD. The final mesh is then obtained using the winkle shape interpolator and the computed geometry of the coarse mesh.

4. Improving the Skeleton-Driven Deformation Approach

The SDD, a classical method for the basic skin deformation is perhaps the most widely used technique in 3D character animation. This method works first by assigning a set of joints with weights to each vertex in the character. The location of a vertex is then calculated by a weighted combination of the transformation of the influencing joints. Although developing a new SDD method is not our main goal, the way the skin deforms is important in our framework since natural looking cloth shape also requires natural skin shape. There are two requirements which the method should fulfill for this particular use: first, it must overcome the undesirable effect of vertex collapse as shown in Figure 2(a). Second, the method must provide an easy way to compute the local coordinate system for each skin vertex. This is necessary as we want to compute the deformation of the cloth surface in relation to the skin surface. We found that the classical SDD can be greatly improved by replacing the linear combination of the matrices by the matrix operator defined by Alexa [16]. The combination of i matrices M_i with their blending weight w_i is given by

$$M_{SSD} = \bigoplus_i w_i \cdot M_i = e^{\sum_i w_i \log(M_i)}.$$

In the remainder of this paper, we denote the function that returns the SDD matrix of the vertex P by $SDD_{\mathrm{P}}()$, $SDD_{\mathrm{P}}()$ taking the joint angles of the skeleton as input. The SDD position of P is simply given by $M_{SDD}.X_{\mathrm{P},Dress}$, $X_{\mathrm{P},Dress}$ being the position of P at initial character pose (see Ref. [17] for further details on SDD).

Note that the operator is not continuous. It is not defined for a rotation of 2π radians between the matrices to be blended.

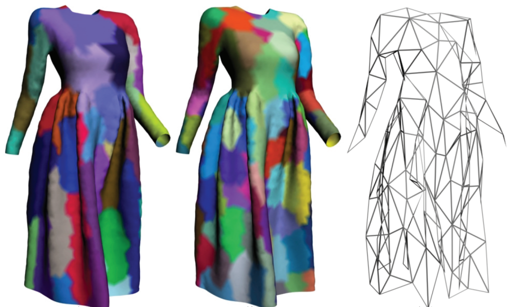

Figure 3: *Two examples of segmentation with (a) patches of large (260 cm^2), and (b) small surface area (95 cm^2). (c) is the coarse mesh corresponding to the segmentation (b). Different colors are randomly assigned to the patches.*

In practice, such case is rare; in general, the largest angle range does not exceed π radians.

5. Simulation of the Gross Behavior

Due to the computational expenses of solving the full numerical system of the physics-based deformation, we seek simplifications by constructing a coarse mesh representation of the garment. The coarse mesh is used to deduce the gross behavior of the cloth in a data-driven manner, based on the input presimulated sequence. A number of optimization strategies are adopted: the two following sections describe a preprocessing that constructs and segments a coarse mesh representation into different region types. We then describe in the next two sections the spring-mass system and collision handling of the coarse mesh at each frame of the simulation. Also described is the runtime process.

5.1. Construction of the Coarse Mesh

We begin by constructing a coarse representation of the given cloth model that will drive the gross behavior of the simulated garment. It consists of two following steps: (1) the cloth surface is partitioned into a set of patches as shown in Figure 3(a) and (b). (2) A coarse mesh representation is obtained by combining a set of vertices in a patch into a single mass point located at the center.

The generation of a patch starts by finding a vertex that has not yet been attributed to a patch that is already generated. The patch is then grown by adding neighboring vertices one after the other. To select a new vertex into the current patch, we evaluate each neighboring vertex that has not been already assigned to a patch, using a penalty function. To enforce the regularity of coarse mesh, which is one condition for obtaining efficient deformation with the mass-spring system [3], we consider two following components.

- Minimizing the 'shape factor': Square Root (Surface Area)/Contour Length. The objective is to obtain 'well-shaped patches', patches that have a circular shape.
- Obtaining patches of equivalent surface area. This component gives a cost that increases with the surface area of the patch. By modifying the significance of this component, we can easily control the number of vertices to be simulated with the physically based deformation (see Figure 3).

The vertex with the lowest cost is selected. When the lowest cost exceeds a threshold, the construction of the patch is completed. We proceed until no vertices can be found to start a new patch. Deciding a good granularity in the coarse mesh is hand-tuned, so that a neat compromise between the simulation quality and the computation load is found. We have found that best simulations are obtained when patch area covers one or two cloth wrinkles.

Note that each patch is associated with a vertex on the coarse mesh. We denote the vector position of a vertex P as X_P, and the vector position of its neighbors as $X_N \in R^{3n}$ (n: the number of neighbors of P).

5.2. Identifying Cloth-to-Joint Relations and Region Types

Next we carry out cloth-to-skin (or body) attachment through skin fitting, by which the skinning data on the cloth mesh are approximated in such a way that the skinning-driven cloth shape best fits the simulated cloth shape throughout the whole presimulated sequence. The basic idea is to use the presimulated results as examples and find the error-minimizing skin data through optimization. An optimization approach, such as the one presented by Mohr and Gleicher [17], could be adopted here. In our case, however, our SDD method is non-linear and therefore the linear regression as adopted by Mohr and Gleicher is not beneficial. Function minimization techniques such as Powell's method [18] can deal with nonlinear functions. Performance is slightly slower, but only preprocessing performance is affected and not runtime performance. The fitting results for a dress model are shown in Figure 4. Notably, the floating regions (colored in red in Figure 4(d)) are attached to the root of the character, as shown in Figure 4(b); this is contributable to the fact that these regions are large in volume and they rarely collide with limbs during the walk motion.

The residual values of the fitting provide useful information on how the garments behave in relation to the body. Intuitively, floating garments such as a skirt, cloth patches may collide with several joints; collisions need to be computed on these regions. On the other hand, the local movements of some cloth patches (such as underwear) are negligible and these patches can be considered as being attached rigidly to the skeleton. In our approach, three regions are identified from

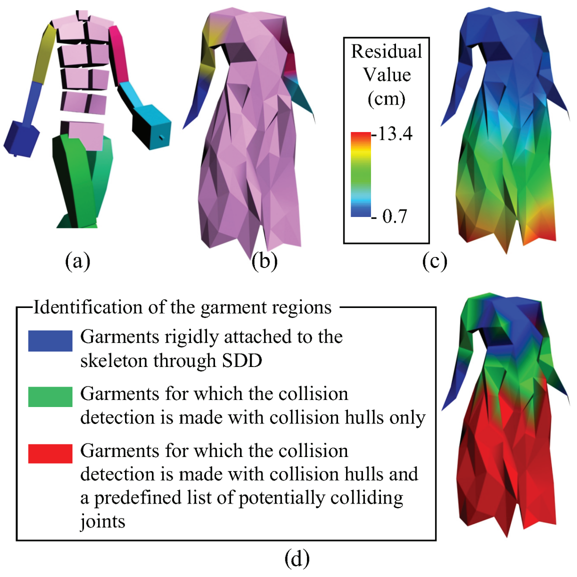
Residual Value (cm)
-13.4
- 0.7
(a)
(b)
(c)
Identification of the garment regions
🟦 Garments rigidly attached to the skeleton through SDD
🟩 Garments for which the collision detection is made with collision hulls only
🟥 Garments for which the collision detection is made with collision hulls and a predefined list of potentially colliding joints
(d)

Figure 4: *(a,b) Influence of the joints on the dress shown in color, (c) quality of the fitting of the SDD data, and (d) the three regions computed on an analysis of the residual values.*

```
1  for each vertex $P \in$ Coarse_Mesh do
2     $M_{SDD} \leftarrow SDD_P$(Skeleton_Joint_Angles)
3     if $P \notin$ Tight_Region
4        $X_P \leftarrow$ Mass_Spring_system(time_step)
5     else
6        $X_P \leftarrow M_{SDD}.X_{D,Dress}$   //Skeleton Driven Deformation
```

Figure 5: *The coarse mesh is computed with both the mass-spring and SDD systems.*

the residual values of the skin fitting process (Figure 4(d)): those that potentially interact with several joints, those that are loosely attached to the skeleton, and those that are rigidly attached to the skeleton. The threshold values are chosen in a way that the coarse mesh deformation remains sufficiently close to the presimulated sequence. For example, a false assignment of loose region into tight region would produce elongated deformations instead of slipping garment over the skin, and therefore generate an overly deformed coarse mesh, which is beyond the training data of the wrinkle generator. Similarly, a false assignment of region 3 into region 2 would result in the garment crossing the legs. In practice, values of 0.5 and 4.0 cm are used to identify tight regions and floating regions, respectively.

The deformation of tight regions is directly computed with the SDD (lines 2 and 6 on Figure 5). The use of SDD for these regions makes it possible to reduce the number of mass points even further. The pseudocode of the simulation loop is given in Figure 5.

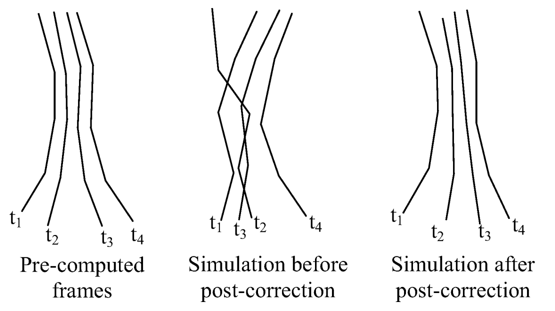
t1
t2
t3
t4
Pre-computed frames
Simulation before post-correction
Simulation after post-correction

Figure 6: *Post-correction of the mass-spring system.*

High residual values indicate much less dependency on a specific body region of the cloth movement. Therefore, an additional collision check is required to handle the interaction of the clothes with the whole body skeleton. A list of potentially colliding body patches is defined by selecting those that approach within a certain distance of the floating regions during the presimulated cloth sequence.

Apart from the position, our SDD computes the local transformation matrix of the vertices, the simulator to be optimized at least for the two following points: limiting collision checks to a small area around the vertices, and the geometric wrinkling which is processed in the SDD local coordinate system.

5.3. Data-Driven Post-Correction of the Coarse Mesh

At each frame of the simulation, we compute the coarse mesh by a mass-spring system with the implicit Euler numerical solver [2]. The simulation run on the coarse mesh hardly reproduces the gross movement of the original cloth because the initial mesh has been significantly simplified (from 4000 to a few dozen vertices) and the topology has been modified. Moreover, unlike the simulator used for the presimulated cloth sequence, the simplified mass-spring model does not accurately simulate the bending and shearing properties of the fabrics [3].

We approach the problem by modifying the behavior of the mass-spring system through a fix-up process (similar to [14]) where the position and velocity of the coarse mesh vertices are modified in order to maintain the cloth shape as close as possible to the original one (Figure 6).

Ideally, the local shape (e.g. position of the vertices in relation to their neighbors) should be a blend of those of the presimulated animation. This is achieved by constructing a set of functions of local shape deformation. Post-correction is accomplished with a function that evaluates the 'ideal' position of the vertex given the position of its neighbors connected by the edges. For each vertex, we construct an interpolating function F_{Post} by using a set of ($X_{N,Presimulated}$, $X_{P,Presimulated}$) pairs extracted from each frame of the presimulated sequence,

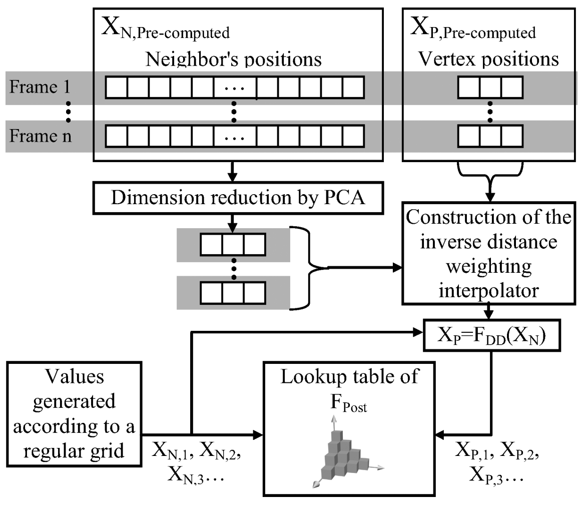
$X_{N,Pre-computed}$
Neighbor's positions
$X_{P,Pre-computed}$
Vertex positions
Frame 1
Frame n
Dimension reduction by PCA
Construction of the inverse distance weighting interpolator
$X_P=F_{DD}(X_N)$
Values generated according to a regular grid
$X_{N,1}$, $X_{N,2}$, $X_{N,3}$...
Lookup table of F_{Post}
$X_{P,1}$, $X_{P,2}$, $X_{P,3}$...

Figure 7: *Construction of the lookup table for the data-driven post-correction.*

```
7   $X'_P \leftarrow X_P$
8   for each vertex $P \in$ Coarse_Mesh do
9     if $P \notin$ Tight_Region
10      $X_P \leftarrow F_{Post}(X'_P, M_{SDD})$
11      $V_P \leftarrow V_P + (X'_P - X_P)$ / time_step
```

Figure 8: *Post-correction of the mass-spring system.*

where $X_N \in R^3$ (n: number of neighbors of P) denotes the position of the neighbors and $X_N \in R^{3n}$ the position of the vertex in question. All these positions are described in the SDD coordinate system of P. The evaluation of the 'ideal' position is made with the inverse distance weighting on the presimulated frames. Given a position of neighbors $X_{N,Input}$ as input, the interpolation computes the corresponding X_P by a weighted summation of the $X_{P,Presimulated}$ values, each weight being computed from the Euclidian distance between $X_{N,Input}$ and all the $X_{N,Presimulated}$ values.

The computation cost of this interpolator grows as the number of presimulated frames increases. We wish to keep the computation cost constant regardless of the duration of the presimulated sequence. A common solution is to construct a lookup table filled with values presimulated by the interpolator on grid sampling (Figure 7). In order to reduce the memory usage of the lookup table, the dimension of $X_{N,Presimulated}$ was reduced prior to the construction of the interpolator, by principal component analysis [18]. The first three principal components, which describe 95% of the average variability of the data, are used.

The positions of the vertices are corrected after every simulation loop. The velocity is updated as well. Its new value is set to the sum of the original velocity and the velocity due to the modification of the vertex position (line 11 on Figure 8).

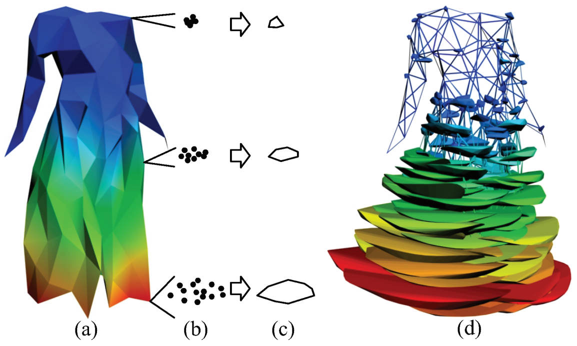

Figure 9: *Computation of the collision hull for each cloth patch: (a) residual values of SDD attachment fitting, (b) computation of local cloth displacement, (c) convex hulls covering all displacements, and (d) resulting collision hulls.*

5.4. Collision Hulls

To prune unnecessary collision tests, we precompute what we term 'collision hulls' that exploit the skin-to-cloth relation obtained from the presimulated sequence. These are built once at the beginning of the simulation (prior to the runtime simulation) after the SDD has been computed on the coarse mesh, using the presimulated sequence. At each presimulated frame, we calculate the difference between the SDD motion model and the presimulated cloth model in the local coordinate system of the SDD. After a sweep, we get a set of points that cover the path a patch takes during the simulation. The smallest convex hull that contains all these points is generated for every patch using the 'Quickhull' algorithm presented by Barber *et al.* [19].

Given enough variation and range of character motion, we expect these hulls to cover the allowable positions of corresponding cloth patches during the runtime simulation. By using collision hulls, collision tests are restricted to a small area around the patch; the overall computation can be significantly reduced in comparison to classical collision detection methods in which collisions are computed between the whole skin and cloth surface. Note that the collision hulls are generated for loose and floating garment regions only. The collision hulls of tight regions are small enough to be approximated by a single point. Figure 9(d) shows the convex hulls computed for the dress model.

Collision handling at runtime consists of correcting the position of coarse mesh vertices after every simulation step so that they remain inside their respective hulls. The algorithm is summarized as follows.

Thus, collision detection returns to computing if the particle is in its associated collision hull (line 14 on Figure 10); the Gilbert–Johnson–Keerthi algorithm [20] is ideally suited to this task. We used constrained dynamics [13] to handle the

```
12 for every vertex P ∈ Coarse_Mesh do
13   if P ∉ Tight_Region
14     if X_p ∉ Collion_Hull(M_SDD)
15       Collision_Response with Collion_Hull
```

Figure 10: *Collision handling using collision hulls (loose and floating regions).*

```
16 for each edge e ∈ Coarse_Mesh do
17   if e ∈ Floating_Region
18     if Collision(e, body_segment)
19       Collision_Response with body_segment
```

Figure 11: *Collision handling on floating regions.*

collision response (i.e. modification of position and velocity in response to collision detection) at line 15. Collision detection is also computed between floating regions and skeleton joints as shown on Figure 11.

5.5. Runtime Computations

The real-time computation of global cloth movements is obtained with a mass-spring system together with the collision response and post-correction described above. The runtime computation of the coarse mesh is obtained in the following order:

- Mass-spring computation (Figure 5).
- Post-correction (Figure 8).
- Collision response on hulls (Figure 10).
- Collision response on floating garments (Figure 11).

Note that the collision response on floating garments comes last to ensure collision avoidance between cloth and body.

6. Generating Garment Details

So far we have shown the first part of our simulation, that is, the coarse-level simulation. We continue now to describe the second part of the simulation, by which detailed cloth shape such as wrinkles or folds are depicted. Again, the main challenge here is obtaining the highest possible realism while maintaining acceptable computation load in order to meet the real-time requirements.

As recognized in earlier works [12,21], wrinkles can be efficiently animated with a geometric method as they are geometric in nature. Unlike previous methods, however, our wrinkling function is not hand-drawn, nor geometrically approximated, but rather trained from on the analysis of the presimulated sequence.

In this work, we choose to represent the wrinkle displacement in the local coordinate system used for SDD. This makes

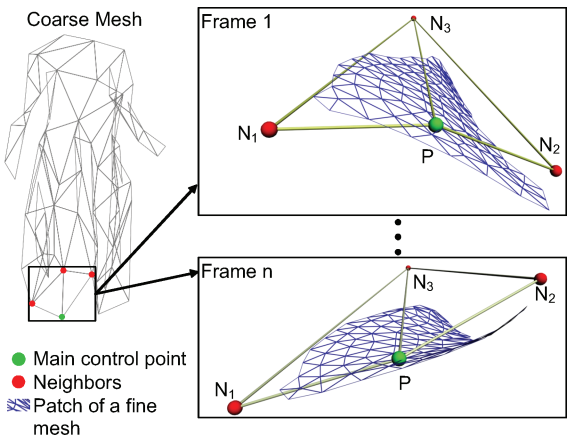

Figure 12: *Shape of the patch with respect to the positions of the control point P and its neighbors N1, N2, and N3.*

our wrinkle parameterization invariant of all joints of higher hierarchy than the currently influencing joint.

Several techniques exist for shape interpolation using examples, such as radial basis functions or parametric interpolation. We have used linear interpolation in which coefficients are defined by multilinear regression on the presimulated animation, since it provides satisfactory results at a very low computation cost. For every vertex x in a patch, the interpolator function takes the associated mass point in the coarse mesh, and its neighbors as input. To calculate the position of x from the input, the wrinkle interpolator interpolates the positions of the coarse mesh points, weighted by coefficients determined the regression model of the following form:

$$F_x(X_{\mathrm{P}}, X_{\mathrm{N}}) = \alpha + \alpha_{\mathrm{P}} X_{\mathrm{P}} + \sum_{n \in \mathrm{Neighbors(P)}} \alpha_n X_n.$$

The values α, α_{P}, and α_{N} are the interpolation coefficients. They are defined by multilinear regression on a set of pairs (positions of coarse mesh vertices, fine mesh vertices) extracted from the presimulated cloth sequence. X_{P} and X_{N} are, respectively, the position of the vertex x and its neighbors; they are all expressed in the SDD coordinate system of x (Figure 12).

Despite its simplicity, linear interpolation works fairly well provided a sufficient number of presimulated frames for the multilinear regression. A condition of a good working interpolator is that the input (i.e. position of the coarse mesh vertices) should be within the range of the presimulated data. In other words, the wrinkle interpolator can only work for the input range for which it has been trained. This condition is maintained thank to the data-driven post-correction (see Section 5.3). This also keeps the smoothness of the boundaries between patches. Figure 13 illustrates the deformation of the wrinkles.

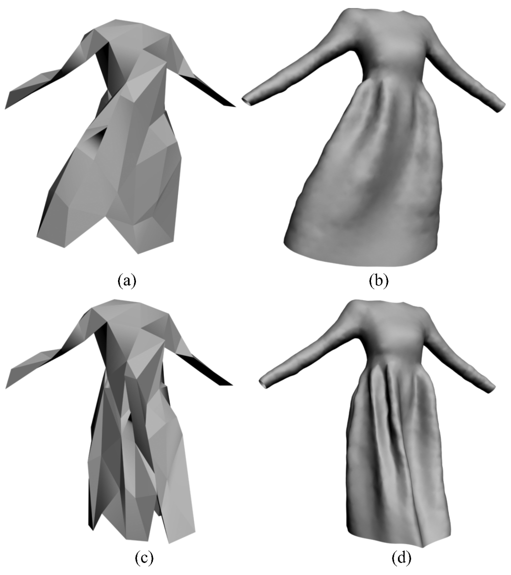

Figure 13: *The wrinkling interpolator in action: wrinkles in (b) and (d) are generated geometrically with (a) and (c) as input.*

7. Results and Discussion

We measure and validate the proposed real-time cloth simulation method along three criteria: the variety of clothes to be simulated, the computation speed, and the range of body motion in the presimulated cloth sequence. Presimulated sequences obtained by the cloth simulator of Baraff *et al.* [3] were used in our preprocessing.

7.1. Variety of Clothes

We used our framework to different types of clothes, as shown on the demonstration video.

- The ‘evening’ dress (Figure 13) is chosen to demonstrate our wrinkle interpolator on large garment regions.
- The ‘cocktail’ dress (Figure 16) is a relatively complex model; the bottom is composed of two layers of tissues and has folds made of large number of vertices, inducing many self-collisions.
- The ‘jeans’ outfit is a good example of a model where the SDD-based geometric approximation can reduce the number of mass points substantially by simulating only a few regions that contribute significantly to the dynamic behavior.

Our simulator behaves fairly well on a wide variety of clothes, including those with highly stiff mechanical properties. Figure 16 show the preprocessing and runtime

Table 2: *Computation speed.*

	Evening Dress	Jeans Outfit	Cocktail Dress
Number of faces	2992	2131	1331
Pre-processing time (min.)	8	8	7
Number of vertices on the rough mesh	110	97	82
Time performance of coarse mesh (fps)	31	63	74
Time performance of the fine mesh (fps)	207	322	588
Overall Performance (fps)	26	51	63

simulation results for the cocktail dress. Moreover, performance will increase due to the fact that the smallest number of triangles will be processed for the real-time rendering.

However, the method may introduce flaws in simulation for some tight clothes due to the approximate handling of collision detection. For some body movements, the skin surface may slightly intersect the cloth surface. Similarly, the same problem may arise for self-collisions on clothes. The deletion of the skin triangles covered by the garment surface can partially correct this drawback.

Note that the cloth simulation is also restricted to clothes worn on bodies. While offering high computation speed, the cloth simulator cannot handle some cloth movements such as those appearing during dressing or undressing. More generally, the clothes are unable to interact with objects other than those that have been taken into consideration during the preprocessing phase. The list of objects that can potentially interact with clothes and the way these objects interact are defined at the preprocessing stage and cannot be changed during the real-time simulation. Finding a method to update the list of possible interacting objects automatically could be a subject for future research.

7.2. Computation speed

Table 2 summarizes the performance of our simulator on a 1 GHz Windows PC. The preprocessing of all the cloth models took less than 10 minutes. All examples run in real-time at approximately 25–50 frames/second (fps), with the coarse mesh deformation process taking about 75% of the total CPU time. As expected, the duration of the presimulated sequence is not a factor of the runtime computation speed. In practice, the performance lowers down at a low rate as the complexity of the collision hulls increases, which tends to be governed by the number of presimulated frames (see Section 5.3).

7.3. Range of body motion

As expected, the quality of the simulation depends on the number and variety of examples—the presimulated sequence in our case. To show that the simulator faithfully recreates the cloth movement used for training, we compared the real-time simulation with the presimulated one in the first

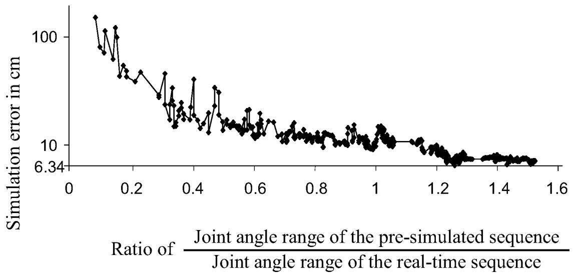

Figure 14: *Estimation of the error when reducing the range of body motion in the pre-simulated sequence.*

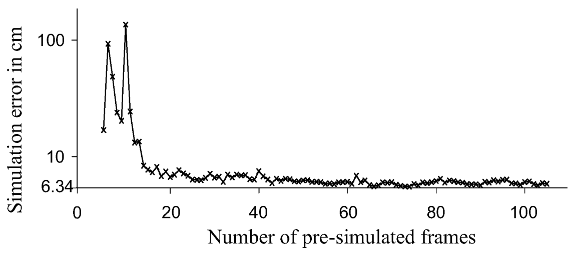

Figure 15: *Estimation of the error when reducing the number pre-simulated frames.*

video. The character walks at a normal pace without any fast movements.

In the second video, different body movements from those of the training were supplied as input to our real-time simulator and the results are compared with the ones generated with a high-quality simulator.

To measure the simulation quality, we compared our simulation results with the presimulated sequence, using a deformation metric. It measures the still shape and movement by the sum of edge length difference and the mass velocity difference over the cloth mesh. Figure 14 shows the importance of the variability of the body motion in the presimulated sequence. The best quality is achieved when the range of the body motion in the presimulated sequence is approximately 30% larger than the one used in the real-time simulation.

Our simulator works well for interpolation (i.e. joint angles within the range of those of the presimulated sequence) but often fails for extrapolation. The main reason for this limitation is collision detection, which does not allow the clothes to have different locations on the body from those calculated in the presimulated sequence; this makes the clothes being attached rigidly to the skeleton.

Figure 15 shows the effect of using motion of different durations (expressed in number of frames) with same joint angle ranges. With less than 70 presimulated frames, the real-time simulation loses its quality.

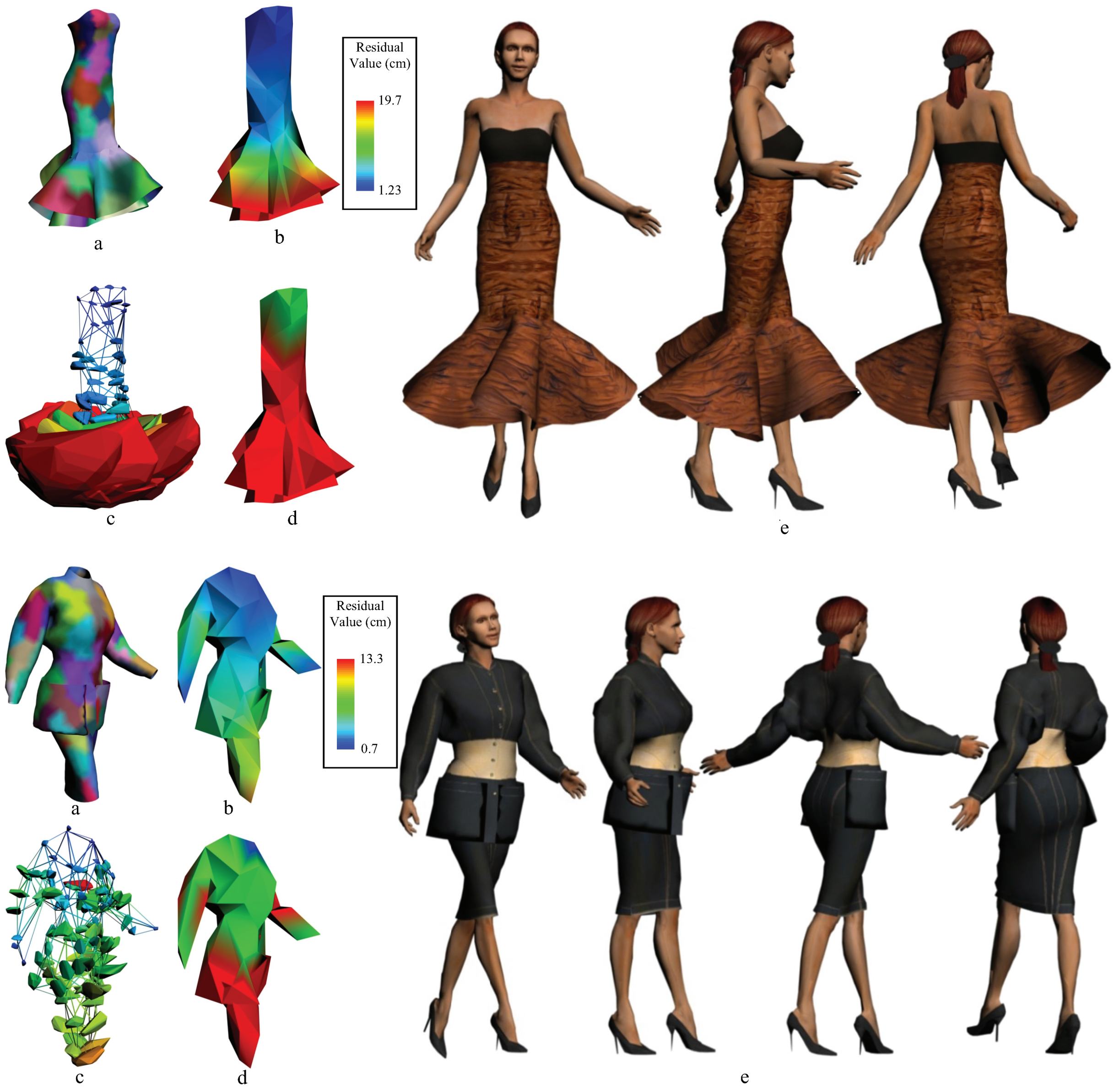

Figure 16: *Simulation of cocktail dress and jean outfit: (a) segmentation, (b) analysis of vertex movements, (c) identification of the three regions, (d) the resulting coarse mesh with collision hulls, and (e) samples of real-time animation.*

8. Conclusion

The recent advent of cloth simulation techniques has matured enough to produce highly realistic cloth movements on animated characters. However, real-time simulation has been largely unexplored until now.

This paper presents the first report of a practical and efficient method for handling real-time simulation almost automatically. We used our framework to produce visually pleasing motion of a wide range of clothes. Both the mass-spring system and collision detection have been rewritten to take advantage of the presimulated sequence of the clothes to be animated. Consequently, our cloth simulator is able to construct a model for real-time animation without user intervention and can deal with different types of clothes from tight to floating with low computation consumption.

There are many interesting avenues for future work. First, the approach could be extended to simulating other physics-based models such as hair and fluid. We also believe that the work on collision hulls is promising. The current mesh model of collision hulls could be replaced by implicit surfaces or voxel maps. Therefore, for a cloth vertex, it could be possible to compute several collisions hulls in relation to different objects in the scene and to compute their intersection for real-time collision detection. By doing so, it may be possible to process collisions on a higher number of objects while maintaining low computation cost. We also believe that the precision of the collision detection could be improved by replacing the convex shape by a surface to follows more closely the trajectories of the vertices.

Acknowledgments

This work is supported by the Swiss National Science Foundation (SNSF). Thanks to Marlène Arévalo and Christiane Luible for their design work to the garment models illustrating this paper.

References

1. K.-J. Choi and H.-S. Ko. Stable but responsive cloth. In *ACM Transactions on Graphics*, *Proceedings of ACM SIGGRAPH 2002*, ACM Press, vol. 21(3), pp. 604–611, 2002.

2. D. Terzopoulos and K. Fleischer. Deformable models. In *The Visual Computer*, Springler-Verlag, vol. 4(6), pp. 306–331, 1988.

3. D. Baraff, A. P. Witkin and M. Kass. Untangling cloth. In *ACM Transaction on Graphics*, ACM Press, vol. 22(3), pp. 862–870, 2003.

4. M. Desbrun, P. Schroder and A. H. Barr. Interactive animation of structured deformable objects. In *Graphics Interface'99 Proceedings*, Morgan Kaufmann, San Mateo, CA, pp. 1–8, 1999.

5. Y.-M. Kang and H.-G. Cho. Bilayered approximate integration for rapid and plausible animation of virtual cloth with realistic wrinkles. In *Computer Animation 2002*, Switzerland. IEEE Press, pp. 203–214, 2002.

6. D. L. James and K. Fatahalian. Precomputing interactive dynamic deformable scenes. In *ACM Transactions on Graphics*, ACM Press, vol. 22(3), pp. 165–172, 2003.

7. F. Cordier and N. Magnenat-Talmann. Real-time animation of dressed virtual humans. In *Eurographics*, Blackwell publishers, vol. 21(3), pp. 327–336, 2002.

8. D. Baraff and A. Witkin. Large steps in cloth simulation. In *ACM Transactions on Graphics,Proceedings of ACM SIGGRAPH*, ACM Press, pp. 43–54, 1998.

9. Z. Kacic-Alesic, M. Nordenstam and D. Bullock. A practical dynamics system. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM Press, pp. 7–16, 2003.

10. X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface'95 Proceedings*, AK Peters, pp. 147–154, 1995.

11. T. Vassilev and B. Spanlang. Fast cloth animation on walking avatars. In *Eurographics*, Blackwell Publishers, Oxford, United Kingdom vol. 20(3), pp. 260–267, 2001.

12. S. Hadap, E. Bangarter, P. Volino and N. Magnenat-Talmann. Animating wrinkles on clothes. In *IEEE Visualization '99*, San Francisco, USA. IEEE Press, pp. 175–182, 1999.

13. R. Bridson, R. Fedkiw and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics*, ACM Press, vol. 21(3), pp. 594–603, 2003.

14. M. Meyer, G. Debuune, M. Desbrun and A. H. Barr. Interactive animation of cloth-like objects in virtual reality. *Journal of Visualization and Computer Animation* 12(1):1–12, John Wiley & Sons, 2001.

15. D. Zhang and M. Yuen. "A Coherence-based Collision Detection Method for Dressed Human Simulation". *Computer Graphics Forum*, Blackwell Publishers, Vol. 21(1), pp. 33–42, 2002.

16. M. Alexa. Linear combination of transformations. In *SIGGRAPH 2002 Conference Proceedings*, Annual Conference Series, ACM Press, vol. 21(3), pp. 380–387, 2002.

17. A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. In *ACM Transactions on Graphics*, ACM Press, vol. 22(3), pp. 165–172, 2003.

18. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling. Numerical recipes in C. In *The Art of Scientific Computing*, Cambridge University Press, pp. 412–420, 1988.

19. C. B. Barber, D. P. Dobkin and H. T. Huhdanpaa. The Quickhull algorithm for convex hulls. In *ACM Transactions on Mathematical Software*, ACM Press, vol. 22(4), pp. 469–483, 1996.

20. E. G. Gilbert, D. W. Johnson and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation* 4(2):193–203, IEEE Press, 1988.

21. Y.-M. Kang, J.-H. Choi, H.-G. Cho and D.-H. Lee. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer Journal* 17(3):147–157, Spinger-Verlag, 2001.

22. P. Volino and N. Magnenat-Thalmann. "Virtual Clothing-Theory and practice", *Springer Verlag*, ISBN: 3-54067-600-7, 2000.