In Proceedings of The Fourth International Conference on Motion in Games (MIG), 2011

Directional Constraint Enforcement for Fast Cloth Simulation

Oktar Ozgen and Marcelo Kallmann

University of California, Merced
Merced, CA, 95343, USA
{oozgen,mkallmann}@ucmerced.edu
http://graphics.ucmerced.edu

Abstract. We introduce a new method that greatly improves the iterative edge length constraint enforcement frequently used in real-time cloth simulation systems for preventing overstretching. Our method is based on the directional enforcement of constraints and on the simultaneous progressive scanning of the cloth edges, starting from fixed vertices and propagating on the direction of gravity. The proposed method successfully detects the meaningful springs to be corrected and ignores the ones that do not have any significance on the overall visual result. The proposed approach is simple and robust and is able to achieve realistic cloth simulations without overstretching, without causing any visual artifacts, and dramatically decreasing the computational cost of the constraint enforcement process. [1]

Keywords: Physically Based Simulation, Cloth Simulation, Constraint Enforcement.

1 Introduction

Cloth simulation is widely used in computer graphics and many systems are able to simulate in real time moderately complex cloth models. Among the several possible approaches for achieving deformable surfaces, particle-based systems remain the most popular model for achieving interactive, real-time results. One common undesired behavior in particle-based cloth models is the effect of overstretching, and specific geometric edge length enforcement procedures are commonly employed.

Iterative enforcement of constraints coupled with explicit integration methods is well suited for cloth simulation systems with modest particle numbers. This combination successfully avoids the requirement of solving systems of equations and therefore leads to computationally fast and visually satisfying simulations, suitable for computer games. Although research has been developed to the problem of constraint enforcement in general, the useful iterative geometric edge length enforcement method has not been improved since its introduction by Provot [15]. We present in this paper a new approach that greatly improves this method. Our method is simple to be implemented, robust, and is able to achieve realistic stiff cloth behavior without any visual artifacts and improving the computation time of the regular iterative constraint enforcement by up to 80%.

[1] Accompanying video at http://graphics.ucmerced.edu/projects/11-mig-edgcorr/

The proposed method is based on computing a meaningful one-pass ordering of edge corrections by considering the direction in which stretching occurs most. First, we experimentally determine the correction priorities over the different types of considered springs in order to minimize the stretch effect of the gravitational pull. Then, we employ corrections starting from all the fixed vertices in a given simulation and synchronously expanding towards the bottom of the cloth. We also limit the correction number of each vertex. By prioritizing the correction order of different types of springs and limiting the correction number of all vertices to a small number, our method successfully detects the important springs to be corrected and ignores the ones that do not have any significance on the overall visual result. Our final method is able to dramatically decrease the cost of the iterative constraint enforcement process and at the same time successfully eliminates undesired overstretching effects.

2 Related Work

Cloth simulation is a central topic in computer graphics and there is a rich literature available [5, 4, 15, 8, 21, 3, 10, 6, 19, 18]. We focus our review on how constraint enforcement has been used in previous works.

A common approach for the iterative constraint enforcement procedure is proposed by Provot [15]. In this method, the cloth is scanned in an iterative way and all springs that are disturbed more than a given threshold in respect to their rest lengths are identified. These springs are then restored to their rest length by moving the two particles connected to a spring along the spring axis, bringing them closer if the spring is over stretched; or further away from each other if the spring is over compressed (see Fig. 1).

The result of the described procedure is that a stiffer cloth behavior is produced. Note however that not all springs can be perfectly corrected since one spring correction can alter a previously corrected spring. There are a number of additional limitations inherent to this kind of brute-force correction approach. One corrective iteration over the cloth does not always guarantee a visually satisfying result, and in addition, the cloth might exhibit an oscillatory behavior in several areas. To overcome these limitations, a common approach is to apply the corrective procedure several times until a visually satisfying result can be obtained. Obviously, increasing the number of constraint enforcement iterations will decrease the performance of the simulation and there is no reliable way to determine how many iterations would be necessary to achieve the desired results. In [15] it is mentioned that the order of correction of springs depends entirely on their used data structure and in cases where constraints are globally extending to the whole cloth object, the order of springs would probably have more importance and should be studied. This observation is the primary motivation for our proposed directional constraint enforcement method.

Aside from iterative approaches, it is also possible to address the problem by solving a system of non-linear constraint equations. Terzopoulos et al [19] used the Gauss-Seidel method to approximate the solution with a linear system of equations [19]. Another popular method is called the Reduced Coordinate Formulation. In this formulation, an unconstrained system with a given number of degrees of freedom is subject to a set of constraints that remove some of the degrees of freedom and parameterizes

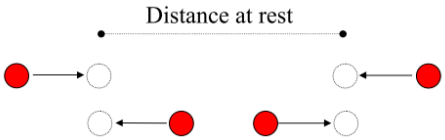

Fig. 1: *Direction of correction for two particles connected by a spring.*

the rest; thus leading to generalized coordinates [11]. Lagrange multiplier formulations are also widely used and have the advantage of defining the velocity and other types of constraints which cannot be formulated using the Reduced Coordinates method [1, 2].

Among these several approaches, our proposed method is most suitable for applications where speed of computation and simplicity of implementation are most important, such as in computer games and real-time simulations.

3 Cloth Model

The mesh structure of our cloth model is based on the scheme presented by [6]. The cloth is represented as a quadrilateral mesh of particles. Particles are connected to each other by massless springs. There are three types of spring elements, which are responsible for the stretch, shear and bend forces. The connectivity of the springs is described in the following way: a particle indexed by $p(i, j)$ is connected by stretch springs to its neighbor particles indexed by $p(i \pm 1, j)$, $p(i, j \pm 1)$. The shear springs connect the particle to $p(i \pm 1, j \pm 1)$, and the bend springs connect the particle to $p(i \pm 2, j)$, $p(i, j \pm 2)$ and $p(i \pm 2, j \pm 2)$.

3.1 Forces

The dynamics of the system is governed by Newton's second law of motion $\boldsymbol{F} = m\boldsymbol{a}$, where m is the mass and $\boldsymbol{a}$ is the acceleration of a particle. The acceleration is computed at every time step based on the total force $\boldsymbol{F}$ applied to the particle, and which accounts for all external and internal forces.

The gravitational force is given by $\boldsymbol{F}_g = m\,\boldsymbol{g}$ and it is the only external force in our system. The internal forces acting on a particle are the forces which originate from the stretch, bend and shear springs connected to each particle. The spring force $\boldsymbol{F}_s$ between two particles is given by Hooke's law:

$$\boldsymbol{F}_s = -k_s(|\boldsymbol{l}| - r)\frac{\boldsymbol{l}}{|\boldsymbol{l}|} \tag{1}$$

where $\boldsymbol{l} = \boldsymbol{x_i} - \boldsymbol{x_j}$ is the difference between the positions of the two particles, r is the rest length of the spring and k_s is the linear spring constant.

3.2 Verlet Integration

Although it is possible to couple the constraint enforcement process with various integration methods, we use the verlet integration method in this work. The Verlet integration is a numerical method that originated from the field of molecular dynamics. Thanks to its simplicity, performance and stability, it has become popular for real-time cloth simulation, in particular in computer games.

The Verlet method stores the current and previous positions of each particle as the state of the system. The velocity is thus implicitly represented by positions. The position update step for each particle is computed with:

$$\boldsymbol{x}_{t+1} = 2\boldsymbol{x}_t - \boldsymbol{x}_{t-1} + \boldsymbol{a}h^2, \tag{2}$$

where $\boldsymbol{x}_{t+1}$, $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$ are the positions of a particle at the next, current and previous timesteps, $\boldsymbol{a}$ is the current acceleration influencing the particle and h is the integration timestep. The acceleration influencing the particle is calculated based on the total force acting on it. The damping of the system might be fine-tuned by changing the constant multiplier (of value 2) in the equation. Decreasing the value to less than 2 will increase the damping; whereas increasing to more than 2 will decrease the damping.

Because the velocity is implicitly calculated, the method tends to stay stable even when relatively large timesteps are used. However, although the simulation remains fast and stable, the use of large timesteps will lead to overshooting of positions and result in a super-elastic cloth. Therefore, simulations relying on Verlet integration are often coupled with an iterative constraint enforcement process that attempts to restore the original distances between each pair of particles connected by a spring in the cloth model.

4 Directional Constraint Enforcement

Our directional constraint enforcement method achieves improved results with a meaningful traversal order of spring corrections. Our method minimizes the number of total corrections needed by preventing redundant repetition of corrections and avoiding the oscillation behavior that often results from a simple iterative constraint enforcement traversal.

Our method is based on the observation that the most stretched springs tend to be the ones on the regions most influenced by the contact forces in a given simulation scenario. Therefore, the most stretched regions on the surface of the cloth are expected to be the regions close to fixed vertices of the cloth, which are vertices that tend not to move relative to their respective colliding object. By defining a traversal order that first corrects the springs around the fixed vertices of the cloth, and expanding the corrections on the direction of the gravity, it is possible to ensure that corrections will be effective and without redundancy.

Our algorithm assumes that the cloth stretches under gravity and has a set of fixed vertices. These fixed vertices correspond to vertices that attach the cloth to a character's body, a flag to a flag pole or a table cloth to the table, etc. Since there may possibly be

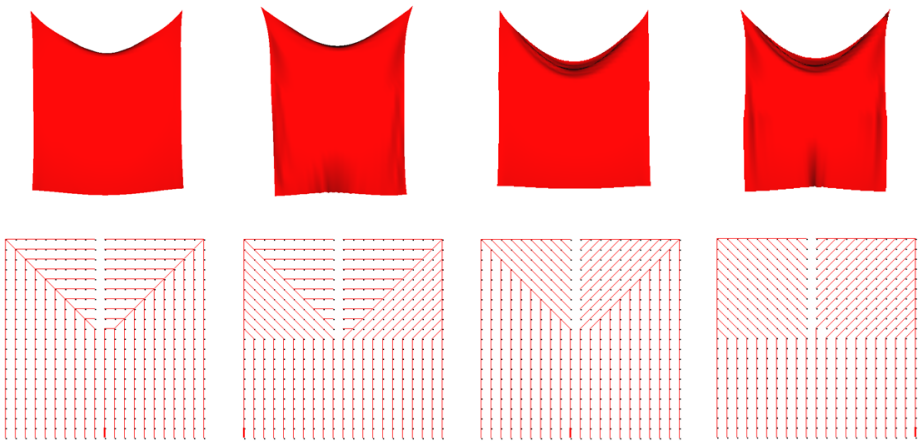

Fig. 2: *Correction patterns and their respective deformation results obtained from choosing different correction orders among the different types of springs.*

more than one fixed vertex to start the correction expansion, it is important to synchronize the corrections originating from different fixed vertices. Our experiments showed that failing to synchronize the expansions are causing serious visual artifacts; and the simultaneous expansion remains an important aspect of the algorithm.

It is also crucial to analyze the overall visual look of the cloth according to the order chosen on the different types of springs. If we categorize the different types of springs along with their elongation directions, we can come up with three major categories: 1) stretch springs elongating towards left or right (horizontal stretch springs), 2) stretch springs elongating towards the bottom of the cloth (vertical stretch springs) and 3) shear springs elongating towards the bottom of the cloth. Note that we do not take into account springs elongating upwards, to make sure that the correction will be expanded towards the gravity direction.

It is not straightforward to determine which type of spring should be given correction priority over other types. We have experimented with different orderings for the three different types of springs, leading to six different combinations. Two of these combinations produced similar correction orders as others; so, we ended up having four different correction orders possible. The correction maps of these four different orderings and their associated deformation results are shown in Fig. 2. The experiment of switching among the different orderings to observe the resulting cloth postures can be found on the accompanying video of this paper. This experiment showed that the best-looking deformations are obtained with the following order of correction: 1) Horizontal stretch springs, 2) Vertical stretch springs, 3) Shear springs.

With the ordering for processing the different types of springs determined, an edge traversal process respecing this ordering can then be devised. Algorithm 1 summarizes our final edge correction ordering generation process. Details of the algorithm are given in the next paragraphs.

Algorithm 1 receives as input the set of fixed vertices V_f. At the beginning of the algorithm, we use an empty queue Q to maintain the synchronous order of correction expansions. When the algorithm terminates, list L will be filled with the right order of edges to be corrected during run time.

The overall algorithm proceeds as follows: first, all the fixed vertices are pushed into the expansion queue Q. Then, until there is no vertex left in the queue, the vertices are popped and expanded one by one. Procedure *Get Horizontal Stretch Springs* (v_s) returns the horizontal stretch springs connected to the vertex v_s. Similarly, *Get Vertical Stretch Springs* (v_s) returns the vertical stretch springs connected to the vertex v_s, and *Get Shear Springs* (v_s) returns the shear springs connected to the vertex v_s. Finally, procedure *Get Goal Vertex* (v_s, s) returns the vertex that is connected to the vertex v_s by the spring s.

Algorithm 1 Ordered Edge Correction

Compute_Correction_List (V_f)

1. $L \leftarrow null$
2. **for all** v such that $v \in V_f$ **do**
3. $\quad Q.push(v)$
4. **end for**
5. **while** Q is not empty **do**
6. $\quad v_S \leftarrow Q.pop()$
7. $\quad S_h \leftarrow$ *Get Horizontal Stretch Springs* (v_s)
8. $\quad$ **for all** s such that $s \in S_h$ **do**
9. $\quad\quad v_g \leftarrow$ *Get Goal Vertex* (v_s, s)
10. $\quad\quad$ *Expand* (L, v_s, v_g)
11. $\quad$ **end for**
12. $\quad S_v \leftarrow$ *Get Vertical Stretch Springs* (v_s)
13. $\quad$ **for all** s such that $s \in V_s$ **do**
14. $\quad\quad v_g \leftarrow$ *Get Goal Vertex* (v_s, s)
15. $\quad\quad$ *Expand* (L, v_s, v_g)
16. $\quad$ **end for**
17. $\quad S_s \leftarrow$ *Get Shear Springs* (v_s)
18. $\quad$ **for all** s such that $s \in S_s$ **do**
19. $\quad\quad v_g \leftarrow$ *Get Goal Vertex* (v_s, s)
20. $\quad\quad$ *Expand* (L, v_s, v_g)
21. $\quad$ **end for**
22. **end while**
23. **return** L

For each vertex expanded we first find all the horizontal stretch springs connected to it. Then, we find the goal vertices that the vertex is connected to through these springs. Finally, we try to expand the correction map towards those vertices. When the *Expand* procedure is called, the expansion to the goal vertex is not guaranteed. The algorithm first checks how many times the goal vertex has been visited. If it already reached the visit limit number then we do not expand. If the visit limit is not reached yet, the algorithm expands to the goal vertex and adds the edge from the current vertex to the

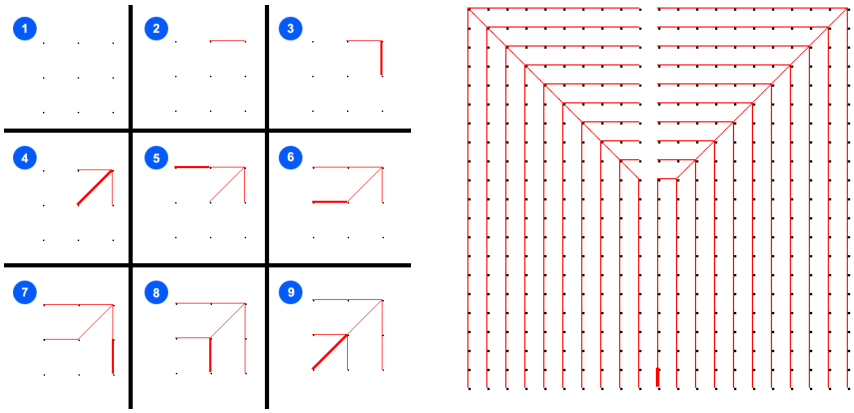

Fig. 3: *Left: step-by-step order of correction starting from an upper right fixed vertex of the cloth. Right: correction map created for a cloth fixed from the two top corners.*

goal vertex to the ordered correction list L. The visit count of the goal vertex will be incremented and the goal vertex will be added to the expansion queue Q. These procedures will be repeated for vertical stretch springs and shear springs as well. The algorithm will stop when there is no vertex left in the queue. At the end of the algorithm, we will have the ordered correction list L completed.

List L is precomputed in advance for each set of fixed vertices to be considered during the real-time simulation. Then, after each integration step in the simulation, the iterative constraint enforcement step will traverse the edges in L, correcting only the position of the second vertices in each edge, satisfying the desired length of each edge. Example results obtained with the algorithm are shown in Figure 3.

4.1 Using multiple correction maps

Our method relies on precomputed fixed orderings (encoded in list L) in order to maximize computation performance in real time. Since the set of fixed vertices may change during a simulation, the validity of the precomputed list L may be reduced depending on how irregular and unpredictable the deformation events are. We tested our method on various scenes such as a planar cloth interacting with a solid ball and a complex cloth model interacting with a women model performing a “catwalk”. At both simple and complex scenarios, the obtained varied cloth-object collisions and cloth-cloth collisions did not seem to cause any significant degradation in the obtained quality of results.

It is also possible to precompute several correction maps to further increase the performance and the accuracy of the animation. Precomputed maps corresponding to different collision states that repeatedly occur in the animation can be used by switching between them according to events in the simulation. For example, in the animation of the walking character wearing a long skirt, the character’s knees are interchangeably colliding with the skirt generating a visible event with impact on the edge correction list.

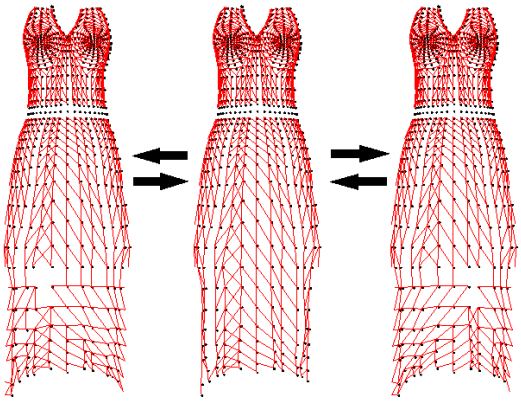

Fig. 4: *Three alternating correction maps used for the long skirt simulation during walking: map with right knee collision (left), map with no knee collision (center), and map with left knee collision (right).*

Everytime a knee collides with the skirt, the colliding vertices are going to be corrected by the collision detection module; we can as well treat them as fixed vertices in additional correction lists. We have obtained improved results with three correction maps for the cases where 1) the knees do not touch the skirt, 2) only the right knee touches the skirt and 3) only the left knee touches the skirt. As shown in Figure 4, everytime we detect a knee collision (or obsence of collision) we switch to its corresponding correction map. If the deformation is specific for one given walking animation, such events can be encoded in the time-parameterization of the animation, eliminating decisions based on collision detection events. The use of multiple correction maps is therefore suitable for simulations dependable on cyclic animations such as walking.

5 Results and Discussion

In order to test the applicability of our method to different cloth simulation scenarios we have tested our directional correction algorithm with clothes with different number of fixed vertices and at different places. Figure 5 shows the different correction maps produced for different initial sets of fixed vertices. Although the order of the corrections change, the number of springs to be corrected stays practically the same. The only exception is the scenario with the falling cloth on the table. In that scenario, a large circular set of fixed vertices is naturally marked as fixed, and the number of springs to be corrected is much less.

We have also tested our method on a more complex long skirt cloth model interacting with a walking character. Our method proved to be efficient and did not cause any undesired visual artifacts. Additional examples of edge correction orders obtained for the skirt cloth model are shown in Figure 6 and rendered results are shown in Figure 7.

Our correction methodology is dependent on the set of fixed vertices on the surface of the cloth. In most of the cloth simulation scenarios, such as hanging clothes, capes,

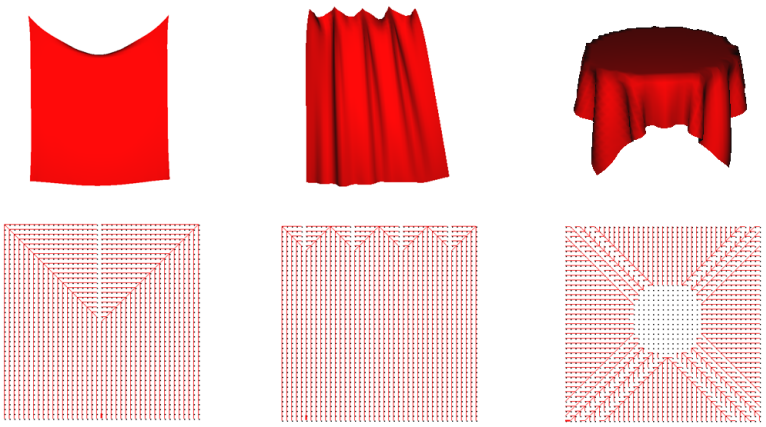

Fig. 5: *The figure shows the correction maps generated by our algorithm and their associated cloth simulation scenarios.*

flags, table clothes, or dressed characters, the set of fixed vertices stay the same during the whole animation. Therefore, in this cases the correction map would only be precomputed once and in a very small amount of time. The precomputation time of the correction map is about 0.52 seconds for a cloth of 1681 particles on a 2.4 GHz Intel Core 2 Duo computer.

In terms of visual quality our model achieves visually satisfying results from a single correction pass over the list of edges. This is a major improvement in respect to the original (non-ordered) constraint enforcement procedure that relies on several passes to achieve good results. One aditional important advantage of our method is that the directional correction completely eliminates the shaky cloth behavior that is frequently seen in multiple pass procedures.

5.1 Limitations

As previously stated, our correction methodology is dependent on how predictable the set of fixed vertices is. In cloth simulation scenarios where the set of fixed points change very frequently, our method would need to update the correction map after each significant change. In such situations an automatic procedure for detecting the validity of correction list L and triggering an ordering update everytime the set of fixed vertices significantly changes from the original set can be integrated in the simulation system. However, this would still maintain the negative effect of penalizing the overall performance of the simulation. On the other hand, if the change on the set of fixed vertices is not frequent, this will not represent a major slow down in the overall computation time.

We are also assuming that the cloth stretches under one single major constant external force (gravity). Although this assumption addresses the vast majority of cloth simulation scenarios, more complex situations where the stretch direction would be affected by multiple varying external forces have not been tested.

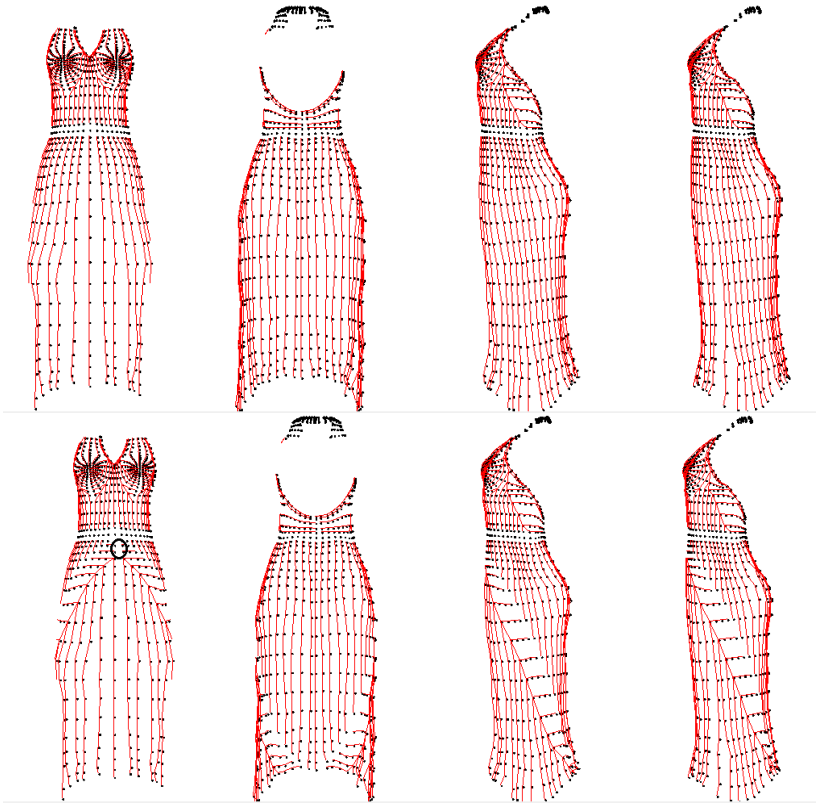

Fig. 6: *Top row: Order of correction obtained with fixed vertices selected at the shoulder and belt regions. Bottom row: A small number of vertices (shown by the circle on the bottom-left image) is added to the set of fixed vertices. Notice how the new correction order obtained is different and how the correction map visually corresponds to the expected buckling of the cloth.*

6 Conclusion

We have introduced a new methodology for determining an efficient and effective traversal order for the iterative edge length enforcement in cloth simulations. Our method is robust, simple to be implemented and is able to achieve realistic stiff cloth behavior without causing overstretching or visual artifacts (like shaky effects). Furthermore, our method improves the computation time by 80 % in respect to the regular iterative constraint enforcement procedure. We believe that our directional constraint enforcement approach will prove itself useful to a number of scenarios employing particle-based deformable models, and in particular in real-time applications such as computer games.

Acknowledgments: We would like to thank Robert Backman for the valuable help with the scene rendering and with the preparation of the accompanying video.

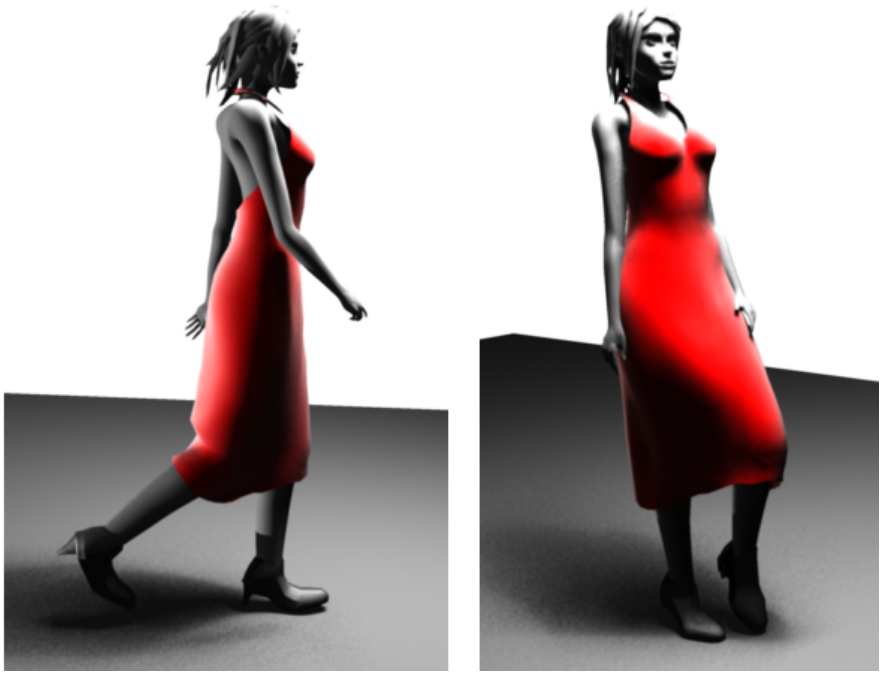

Fig. 7: *Example of a long skirt that would significantly overstretch if no constraint enforcement is made. Our method eliminates the overstreching in an efficient manner.*

References

1. Baraff, D.: Linear-time dynamics using lagrange multipliers. In: Computer Graphcis Proceedings, Annual Conference Series. pp. 137–146. New York, NY, USA (1996)
2. Baraff, D., Witkin, A.: Physically based modeling: Principles and practice. In: ACM SIGGRAPH '97 Course Notes (1997)
3. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: Proc. of SIGGRAPH'98. pp. 43–54. New York, NY, USA (1998)
4. Breen, D., House, D., Wozny, M.: Predicting the drape of woven cloth using interacting particles. In: Proc. of SIGGRAPH'94. pp. 365–372. ACM, New York, NY, USA (1992)
5. Carignan, M., Yang, Y., Magenenat-Thalmann, N., Thalmann, D.: Dressing animated synthetic actors with complex deformable clothes. In: Proc. of SIGGRAPH'92. pp. 99–104. ACM, New York, NY, USA (1992)
6. Choi, K., Ko, H.: Stable but responsive cloth. In: Proc. of SIGGRAPH'02. pp. 604–611. ACM, New York, NY, USA (2002)
7. Desbrun, M., Schröder, P., Barr, A.: Interactive animation of structured deformable objects. In: Proc. of Graphics interface. pp. 1–8. San Francisco, CA, USA (1999)
8. Eberhardt, B., Weber, A., Strasser, W.: A fast, flexible, particle-system model for cloth draping. In: IEEE Computer Graphics and Applications'96. pp. 52–59. IEEE (1996)
9. Hauth, M., Etzmuß, O., Straßer, W.: Analysis of numerical methods for the simulation of deformable models. The Visual Computer 19(7-8), 581–600 (2003)

10. House, D.H., Breen, D.E. (eds.): Cloth modeling and animation. A. K. Peters, Ltd., Natick, MA, USA (2000)
11. Isaacs, P.M., Cohen, M.F.: Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. In: Computer Graphcis Proceedings, Annual Conference Series. pp. 131–140. New York, NY, USA (1987)
12. Ling, L.: Aerodynamic effects. In: House, D.H., Breen, D.E. (eds.) Cloth modeling and animation, pp. 175,195. A. K. Peters, Ltd., Natick, MA, USA (2000)
13. Ling, L., Damodaran, M., Gay, R.K.L.: Aerodynamic force models for animating cloth motion in air flow. The Visual Computer 12(2), 84–104 (1996)
14. Müller, M., Schirm, S., Teschner, M., Heidelberger, B., Gross, M.: Interaction of fluids with deformable solids: Research articles. Computer Animation and Virtual Worlds 15(3-4), 159–171 (2004)
15. Provot, X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In: Proc. of Graphics Interface '95. pp. 147–155. New York, NY, USA (1995)
16. Robinson-Mosher, A., Shinar, T., Gretarsson, J., Su, J., Fedkiw, R.: Two-way coupling of fluids to rigid and deformable solids and shells. ACM Transactions on Graphics (Proc. of SIGGRAPH'08) 27(3), 1–9 (2008)
17. Selle, A., Su, J., Irving, G., Fedkiw, R.: Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. Transactions on Visualization and Computer Graphics 15(2) (2009)
18. Terzopoulos, D., Fleischer, K.: Deformable models. The Visual Computer 4(6), 306–331 (1988)
19. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. In: Proc. of SIGGRAPH '87. pp. 205–214. ACM, New York, NY, USA (1987)
20. Tu, X., Terzopoulos, D.: Artificial fishes: physics, locomotion, perception, behavior. In: Proc. of SIGGRAPH '94. pp. 43–50. ACM Press, New York, NY, USA (1994)
21. Volino, P., Courchesne, M., Thalmann, N.M.: Versatile and efficient techniques for simulating cloth and other deformable objects. In: Proc. of SIGGRAPH'95. pp. 137–144. ACM, New York, NY, USA (1995)