Plausible Real-time Cloth Animation

Young-Min Kang and Hwan-Gue Cho

Department of Computer Science, Pusan National University, Jangjeon-dong, Keumjeong-gu, Pusan, 609-735, KOREA. {ymkang,hgcho}@pearl.cs.pusan.ac.kr

In this paper, we present a real-time animation technique for deformable objects based on mass-spring models. Although many researchers have proposed various techniques for representing the motion and the appearance of deformable objects, the real-time animation of deformable object is still a hard problem. The difficulties in cloth animation mainly lie in the fact that cloth simulation easily tends to become unstable. There have been a few efficient solutions for real-time animation [3,5]. However, the previous techniques for real-time cloth animation are not capable of generating the plausible motion and appearance because the methods are based on excessive approximations. This paper proposes an efficient technique that can generate realistic animation of complex cloth objects, and the technique makes it possible to integrate realistic deformable objects into the virtual reality systems without violating the interactivity of the system. The proposed method is based on the implicit integration in order to keep the system stable, and the solution of the linear system involved in the implicit integration is efficiently approximated in real-time.

Keywords: *Virtual reality, Interactivity, Cloth animation*

1. Introduction

Many researchers have endeavored to devise efficient methods for cloth simulation or animation, and one of the most important advances in recent years is the use of implicit integration with large steps [1]. However, the implicit integration method requires the solution of a linear system, which involves a large matrix with a size proportional to the square of the number of mass-points. Therefore, it is impossible to generate real-time animation of cloth model even with the implicit method when the number of mass-points is sufficiently large enough to represent realistic wrinkles.

Some researchers have tried to devise efficient methods that can interactively manipulate the deformable objects in virtual environments, and a few works have been proposed to surmount the limitation of the implicit method [3,5]. The previous techniques for real-time animation sacrificed the accuracy in order to achieve real-time or interactive performance. Desbrun *et al.* proposed an efficient technique that approximates the matrix involved in the linear system as a constant matrix and pre-computes the inverse matrix [3]. They applied the pre-computed inverse matrix as a force filter at every time-step in order to calculate the states of the cloth model at interactive rates. Although this method is more efficient than the original implicit method, it also involves $O(n^2)$-sized matrix where n is the number of mass-points in the mass-spring model, and the pre-computed inverse matrix is usually a dense matrix. A stable method that approximates the solution with a direct update formula has been also introduced. No matrix operations are involved in the method so that the method can update the state of the deformable object models in $O(n^2)$ time with time-steps of arbitrary sizes [5]. However, the method excessively damps the motion in order to maintain the stability and efficiency of the computation. Therefore, the method generates slower motion than the real motion of cloth when high stiffness or large time-step is used. Oshita and Makinouchi have proposed a geometric approach that generates wrinkly details on cloth models composed of sparse particles [7]. The method produces smooth cloth-like surfaces based on PN triangulation proposed in [9]. However, the geometric approach cannot produce the realistic appearance of cloth because the geometric interpolation of the initial coarse mesh does not take the physical reality into account.

Immersive virtual reality environments require realistic scene that can interact with user behaviors in real-time. Therefore, the deformable objects in virtual reality systems must be animated in an efficient way, while keeping the structure of the model complex enough to guarantee the plausible appearance.

2. Cloth Animation based on Mass-Spring Model

In this paper, we consider a mass-spring based cloth model, which is the most intuitive and simple model for efficient animation of cloth. The spring force can be easily computed with Hooke's law, and the movement of a particle can be described with Newton's second law. Therefore, the cloth animation is the numerical integration of the force for computing the new velocity and the integration of the new velocity for computing the new position of each mass-point. However, the stability problem disables us to employ a simple explicit integration method. However, the instability can be avoided with implicit methods. Therefore, we can stably animate the mass-spring based objects by integrating the spring forces with the following backward Euler method [3,6,11]:

$$\begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^{t} + h\mathbf{M}^{-1}\mathbf{f}^{t+h} \\ \mathbf{x}^{t} + h\mathbf{v}^{t+h} \end{pmatrix} \tag{1}$$

where h denotes the time interval, and $\mathbf{v}$ denotes the vector of velocity values of the mass-points. Similarly, $\mathbf{f}$ and $\mathbf{x}$ are the vectors of forces and locations of the mass-points respectively. The matrix $\mathbf{M}$ is the mass matrix. The superscripts t and $t+h$ denote time, and semantically mean the current state and the next state respectively. $\mathbf{v}$, $\mathbf{f}$, and $\mathbf{x}$ are vectors, and $\mathbf{M}$ is a matrix. Their elements $\mathbf{f}_i$, $\mathbf{x}_i$, $\mathbf{v}_i$, and m_i denote the force, location, velocity, and mass of the mass-point i respectively.

The objective of computation is to find $\mathbf{x}^{t+h}$ (i.e., the new positions of the mass-points at the next step), and the vector $\mathbf{x}^{t+h}$ can be easily determined when we know $\mathbf{v}^{t+h}$ (the next velocities of the mass-points). The next velocity vector can be computed if we find the velocity change vector $\Delta\mathbf{v}^{t+h}$ that denotes $\mathbf{v}^{t+h}$–$\mathbf{v}^{t}$. Therefore, the cloth animation with Eq. 1 is eventually reduced to finding $\Delta\mathbf{v}^{t+h}$ as follows:

$$\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^{t+h} \tag{2}$$

Although the mass-spring based animation can be described as a simple equation like Eq. 2, the real-time animation is not an easy problem. The difficulty arises from the unknown force vector at time $t+h$. Since Hooke's law can compute only $\mathbf{f}^{t}$, the force at the next time step should be approximated as follows:

$$\mathbf{f}^{t+h} = \mathbf{f}^{t} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Delta\mathbf{x}^{t+h} = \mathbf{f}^{t} + \mathbf{J}\Delta\mathbf{x}^{t+h} \tag{3}$$

where $\mathbf{J}$ denotes the Jacobian matrix of the force vector with respect to the position vector, and $\Delta\mathbf{x}^{t+h}$ is the positional change from time t to $t+h$. The size of the matrix $\mathbf{J}$ is $n \times n$, and each component of $\mathbf{J}$ is a 3×3 submatrix.

Since the positional change $\Delta\mathbf{x}^{t+h}$ can be rewritten as $(\mathbf{v}^{t} + \Delta\mathbf{v}^{t+h})$h, the force at the next state can be expressed as follows:

$$\mathbf{f}^{t+h} = \mathbf{f}^{t} + \mathbf{J}(\mathbf{v}^{t} + \Delta\mathbf{v}^{t+h})h \tag{4}$$

With the force approximation shown in Eq. 4, the animation of the mass-spring model is finally reduced to a linear system solving as follows:

$$\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^{t} + h^2\mathbf{M}^{-1}\mathbf{J}\mathbf{v}^{t} + h^2\mathbf{M}^{-1}\mathbf{J}\Delta\mathbf{v}^{t+h} \tag{5}$$

By multiplying the mass-matrix $\mathbf{M}$ to the both side, the problem can be finally expressed as follows:

$$(\mathbf{M} - h^2\mathbf{J})\Delta\mathbf{v}^{t+h} = h\mathbf{f}^{t} + h^2\mathbf{J}\mathbf{v}^{t} \tag{6}$$

The stable animation of the mass-spring model can be achieved by solving the linear system shown in Eq. 6. However, the stability is not a complete solution to the real-time animation. Since the matrix $\mathbf{M}$–$h^2\mathbf{J}$ is sparse and symmetric, the linear system can be solved efficiently. However, the size of the matrix rapidly increases when the number of mass-points increases. In order to guarantee realistic virtual environments,

sufficiently large number of mass-points must be used. However, the exact solution of Eq. 6 requires too heavy computation while the system is supposed to satisfy certain time constraints.

3. Real-time Animation of Cloth Model

In general, the matrix $\mathbf{M}–h^2\mathbf{J}$ in Eq. 6 is so sparse that iterative methods such as conjugate gradient methods can be successfully employed. Although the sparseness of the matrix does not guarantee the real-time performance of the conventional linear system solvers, it is the most important property. Our real-time animation method also exploits the sparseness of the matrix, and other properties in order to compute the next state as fast as possible.

Let $\mathbf{f}_i^{(i,j)}$ denotes the force on the mass-point *i* caused by the spring between the mass-points *i* and *j*. If the spring force is simply modeled with Hooke's law and κ_{ij} denotes the spring constants between the mass-points, $\mathbf{f}_i^{(i,j)}$ can be described as follows:

$$\mathbf{f}_i^{(i,j)} = \kappa_{ij}(|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0)\frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|} \tag{7}$$

The total spring force on the *i*-th mass-point is then the sum of all the spring forces caused by all the springs linked to the mass-point. Let l^0_{ij} denote the native length of the spring between the *i*-th and *j*-th mass-points. The derivative of force on the mass-point *i* with respect to the location of the mass-point *j* can then be expressed as follows:

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \kappa_{ij}\frac{|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|}\mathbf{I}_3 + \kappa_{ij}\cdot\frac{l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|}\left(\frac{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^{\mathbf{T}}}{|\mathbf{x}_j - \mathbf{x}_i|^2}\right) \tag{8}$$

However, the force derivative in Eq. 8 has serious problem when the distance between the mass-points *i* and *j* is smaller than l^0_{ij}. When the distance between two mass-points approaches to 0, the derivative of the force contains extremely large values as components, and it makes the system fail. In order to avoid the undesirable situations, we approximated the force derivative by assuming l^0_{ij} has the same value as $|\mathbf{x}_j\text{-}\mathbf{x}_i|$. Then the components of the force derivative are guaranteed to have smaller values than the spring constant of the spring. Therefore, the force derivative used for our method was approximated as follows:

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \mathbf{J}_{ij} \cong \kappa_{ij}\left(\frac{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^{\mathbf{T}}}{|\mathbf{x}_j - \mathbf{x}_i|^2}\right) \tag{9}$$

Henceforth, $\mathbf{J}_{ij}$ denotes the approximate force derivative shown in Eq. 9, and $\mathbf{J}_{ii}$ denotes the negated sum of the approximate force derivatives (i.e., $\mathbf{J}_{ii} = -\Sigma_{(i,j)\in E}\mathbf{J}_{ij}$).

Since **M** is a diagonal matrix and the Jacobian matrix **J** is sparse, it is obvious that $\mathbf{M}–h^2\mathbf{J}$ is also sparse. For the simplicity, let us define a matrix **W** as follows:

$$\mathbf{W} = \mathbf{M}–\mathrm{h}^2\mathbf{J} \tag{10}$$

The components of the matrix **W** are 3×3 submatrices, which can be easily computed. The component in the *i*-th row and *j*-th column, $\mathbf{W}_{ij}$, is $–h^2\mathbf{J}_{ij}$ if *i* and *j* are different from each other. The diagonal components, $\mathbf{W}_{ii}$, can be computed as $\mathbf{M}_i–h^2\mathbf{J}_{ii}$. In other words, the *i*-th diagonal component is the sum of the off-diagonal components in the row and mass submatrix of the *i*-th mass-point. Therefore, the diagonal components can be expressed as $\mathbf{M}_i+h^2\Sigma_{(i,j)\in E}\mathbf{J}_{ij}$ where E is the set of springs

It is obvious that the matrix is symmetric because $\mathbf{J}_{ij}$ and $\mathbf{J}_{ji}$ are identical. One additional property is that each submatrix is also a symmetric matrix. The symmetry of the whole matrix and submatrices enables us to efficiently store the matrix, and the inverses of the submatrices can be computed in a more efficient way because they are 3×3 symmetric matrices.

The matrix **W** is sparse because **M** is diagonal matrix and a component of the Jacobian matrix $\mathbf{J}_{ij}$ has a non-zero matrix only when the mass-point *i* and *j* are linked with a spring. Therefore, the number of non-zero components (3×3 submatrices) in the *i*-th row of the matrix is simply one larger than the number of springs linked to the mass-point *i*. In general, only a small number of springs are linked to a specific mass-point, and it is not affected by the complexity of the model, i.e., the number of total mass-points in the cloth model. Therefore, each row in the matrix has a limited number of effective components, and the sparseness of the matrix increases as the number of total mass-points increases.

The vector ($h\mathbf{f}^t+h^2\mathbf{J}\mathrm{v}^t$) in the right side of the Eq. 6 can be described as $h(\mathbf{f}^t+h\mathbf{J}\mathrm{v}^t)$ and the additional forces $h\mathbf{J}\mathrm{v}^t$ can be considered as viscosity forces. Because of the sparseness of the Jacobian matrix, we can efficiently compute the vector $h\mathbf{J}\mathrm{v}^t$. If we denote the sum of spring forces and viscosity forces ($\mathbf{f}^t+h\mathbf{J}\mathrm{v}^t$) as internal forces $\tilde{\mathbf{f}}$, each component (3-dimensional vector) of internal force vector can be calculated with the consideration of linked mass-points as follows:

$$\begin{aligned}\tilde{\mathbf{f}}_i^t &= \mathbf{f}_i^t + h\textstyle\sum_{(i,j)\in E}\mathbf{J}_{ij}\mathbf{v}_j^t + h\mathbf{J}_{ii}\mathbf{v}_i^t \\ &= \mathbf{f}_i^t + h\textstyle\sum_{(i,j)\in E}\mathbf{J}_{ij}(\mathbf{v}_j^t - \mathbf{v}_i^t)\end{aligned} \tag{11}$$

Then, the linear system shown in Eq. 6 can be simply rewritten as follows:

$$\mathbf{W}\Delta\mathbf{v}^{t+h} = h\tilde{\mathbf{f}}^t \tag{12}$$

Due to the properties of the matrix $\mathbf{W}$, the linear system of Eq. 12 can be expressed as following n equations:

$$\begin{aligned}\mathbf{W}_{11}\Delta\mathbf{v}_1^{t+h} &= h\tilde{\mathbf{f}}_1^t + h^2\textstyle\sum_{(1,j)\in E}\mathbf{J}_{1j}\Delta\mathbf{v}_j^{t+h} \\ \mathbf{W}_{22}\Delta\mathbf{v}_2^{t+h} &= h\tilde{\mathbf{f}}_2^t + h^2\textstyle\sum_{(2,j)\in E}\mathbf{J}_{2j}\Delta\mathbf{v}_j^{t+h} \\ &\vdots \\ \mathbf{W}_{nn}\Delta\mathbf{v}_n^{t+h} &= h\tilde{\mathbf{f}}_n^t + h^2\textstyle\sum_{(n,j)\in E}\mathbf{J}_{nj}\Delta\mathbf{v}_j^{t+h}\end{aligned} \tag{13}$$

All the equations in Eq. 13 have a similar form, and the i-th equation can be rearranged to highlight the velocity change of the i-th mass-point as follows:

$$\Delta\mathbf{v}_i^{t+h} = \mathbf{W}_{ii}^{-1}(h\tilde{\mathbf{f}}_i^t + h^2\textstyle\sum_{(i,j)\in E}\mathbf{J}_{ij}\Delta\mathbf{v}_j^{t+h}) \tag{14}$$

The matrices ($\mathbf{M}_i$, $\mathbf{J}_{ii}$, and $\mathbf{J}_{ij}$) involved in Eq. 14 are 3×3 symmetric matrices, and the computation of the vector $\Sigma\mathbf{J}_{ij}\Delta\mathbf{v}_j^{t+h}$ requires a small amount of computations if we know the velocity change of linked mass-points at the next time step $\Delta\mathbf{v}_j^{t+h}$. Therefore, the motion of the cloth model can be computed in $O(n+e)$ time if the velocity change of each mass-point can be computed independently with Eq. 14. However, each equation, unfortunately, cannot be computed independently because it is related with the result of some other equations ($\Delta\mathbf{v}_j^{t+h}$).

Our method approximates the solution of the linear system by iteratively computing the velocity change of each mass-point with Eq. 14. Let $\Delta\mathbf{v}_j^{t+h\ (k)}$ be the result after the k iterations, and suppose that the initial guess of velocity change of each mass-point is simply $\Delta\mathbf{v}_i^{t+h} = \mathbf{W}_{ii}^{-1}h\tilde{\mathbf{f}}_i^t$ by ignoring the velocity changes of linked mass-points. During the iterative update of $\Delta\mathbf{v}_i^{t+h}$, the matrix $\mathbf{M}_i - h^2\mathbf{J}_{ii}$ (i.e., $\mathbf{W}_{ii}$) and the vector $h\tilde{\mathbf{f}}_i$ remain constant. Therefore, these matrix and vector requires only one computation during the iterations for one time-step. Let us denote the constant matrix as $\mathbf{W}_i$ and the constant vector $\mathbf{p}_i$ as follows:

$$\begin{aligned}\mathbf{W}_i &= \mathbf{W}_{ii} = (\mathbf{M}_i - h^2\mathbf{J}_{ii}) &&\in \mathbf{R}^{3\times 3} \\ \mathbf{p}_i &= h\mathbf{W}_i^{-1}\tilde{\mathbf{f}}_i &&\in \mathbf{R}^3\end{aligned} \tag{15}$$

In most cases, the approximate solution after only one iterative update was stable and plausible enough to be used in real-time system. Therefore, we approximated the velocity change of each mass-point as follows:

$$\Delta\mathbf{v}_i^{t+h} \cong \Delta\mathbf{v}_i^{t+h^{(1)}} = \mathbf{p}_i + h^2\mathbf{W}_i^{-1}(\textstyle\sum_{i,j\in E}\mathbf{J}_{ij}\mathbf{p}_j) \tag{16}$$

The fast approximation of the velocity change of each mass-point with Eq. 16 enables real-time animation of deformable objects based on mass-spring models.

4. Experiment

The proposed method has been successfully implemented for an interactive cloth animation system. The cloth animation system has been developed on PC environments. Even with the medium performance level PC, the method could generate real-time animation of virtual cloth with thousands of polygons. Usually, accurate collision handling methods require heavy computations. The collision handling was not a

major subject of our research, and we employed a simplified collision resolution.

Fig. 1 shows the interactive animation system for manipulating cloth. The system enables users to load mesh structures and animate the mesh in real-time. Because of the real-time performance of the system, users can efficiently control the various properties of the cloth model. Therefore, this system can be used as input interface for determining and setting the parameters of the cloth models to be animated.

Fig. 2 shows the result when the proposed method was applied to real-time animation of virtual character with simple cloth model. The character model shown in Fig. 2 interactively moves and the cloth model was animated in accordance with the movement of the character model.

We have tested the method in medium level PC environments (Pentium II 800Mhz), and the proposed method showed real-time performance for the animation of complex models with thousands of polygons. Because the update process for each mass-point considers only neighboring mass-points linked by springs, it is obvious that the time complexity of the proposed method is linearly proportional to the number of mass-points and spring edges. Fig. 3 visually shows the time required to compute the next state according to the number of mass-points and spring edges.

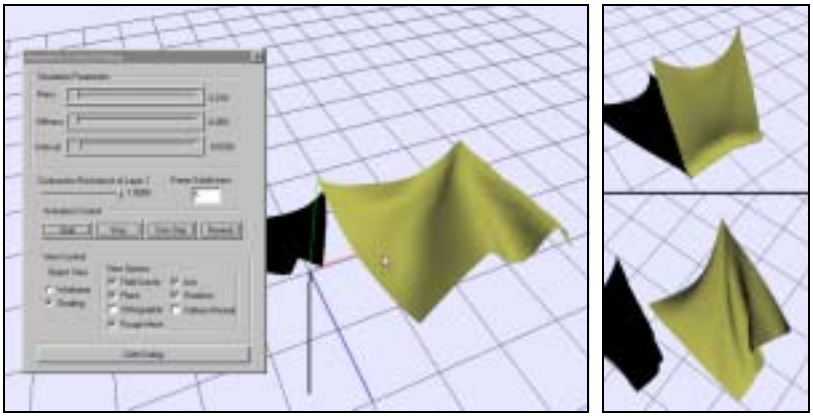

Fig 1. Interactive Cloth Manipulation System

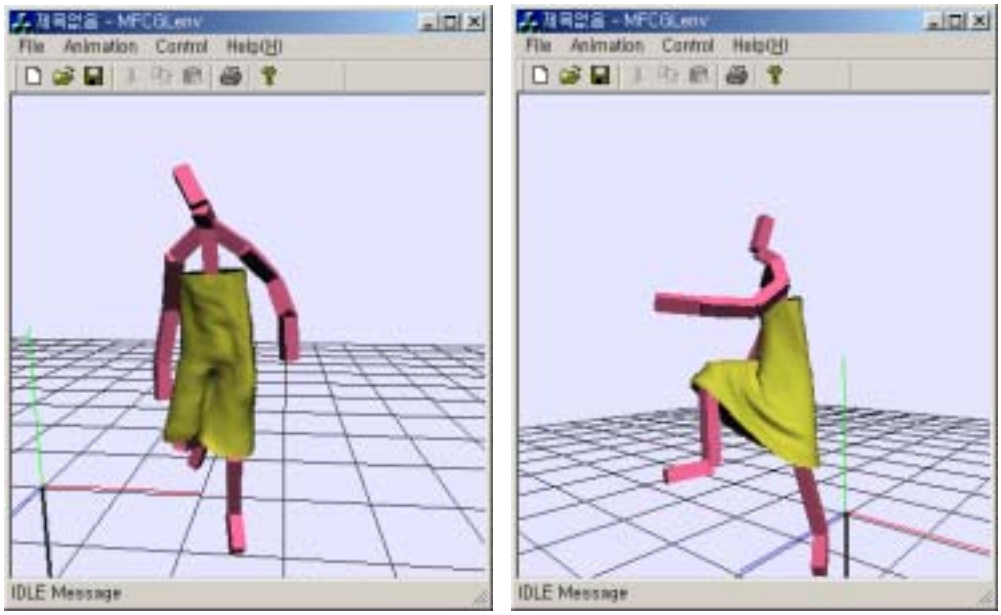

Fig 2. Interactive Animation System with Simple Character Model

5. Conclusion

In this paper, we presented an efficient method to animate deformable objects based on mass-spring models in real-time. The virtual reality environments require stable and efficient real-time animation methods. The instability of a numerical integration method affects the overall performance of the cloth

animation system because the unstable method requires extremely small time-steps. Therefore, real-time animation of a complex cloth model is not feasible with the unstable methods. For that reason, the stability is the first essential property of animation techniques in real-time environments such as VR systems.

The proposed method provides a sufficient stability because the approximate solution of the method is based on the implicit integration. The stability of the method has been obtained by employing the implicit integration method for deriving the iterative state-update scheme that computes the next state of each mass-points. Therefore, the method can produce cloth animation with large time-steps.

Although the stability is the essential property, it is not a sufficient property for the real-time animation. The stability of the implicit method requires heavy computational burdens to obtain the next state of each mass-point because it casts the original problem into a linear system problem. However, the linear system for mass-spring animation derived by the implicit method has many properties, which can be exploited for improving the performance of the animation system. The matrix involved in the linear system is a sparse, symmetric, and block matrix. The proposed method efficiently approximates the solution of the linear system by taking advantage of these properties. The proposed method independently computes the next state of each mass-point by considering only linked mass-points so that the computation of the next state of the whole cloth model can be achieved in O(n+e) time where n is the number of the mass-points and e is the number of the spring edges in the cloth model. Therefore, the efficiency and the stability of the method enable real-time animation of deformable objects to be integrated in virtual environments.

Apart from the real-time performance, the proposed method has additional important advantages. The proposed method is intuitive and easy to implement. The method can be easily implemented with simple modules for inverse computation of 3×3 matrices and multiplication of 3×3 matrices and vectors so that the method does not require developers to struggle with optimization of the storage and the performance of the linear system solver. The easiness of the implementation guarantees that the implementation of the proposed method in various real-time or interactive systems will be successfully achieved with minimum efforts.

References

1) D. Baraff and A. Witkin. “Large steps in cloth simulation”, Proceedings of SIGGRAPH 98, pp. 43-54 (1998).
2) D. E. Breen, D. H. House, and M. J. Wozny. “Predicting the drape of woven cloth using interacting particles”, Proceedings of SIGGRAPH '94, pp. 365-372 (1994).
3) M. Desbrun, P. Schröder, and A. Barr. “Interactive animation of structured deformable objects”, Graphics Interface '99, pp. 1-8 (1999).
4) B. Eberhardt, A. Weber, and W. Strasser. “A fast, flexible particle-system model for cloth draping”, IEEE Computer Graphics & Applications, 16(5):52-59 (1996).
5) Y.-M. Kang, J.-H. Choi, H.-G. Cho, and C.-J. Park. “An efficient animation of wrinkled cloth with approximate implicit integration”, The Visual Computer, 17(3):147-157 (2001).
6) M. Kass. “An introduction to continuum dynamics for computer graphics”, SIGGRAPH Course Note: Physically-based Modeling. ACM SIGGRAPH (1995).
7) M. Oshita and A. Makinouchi. “Real-time cloth simulation with sparse particles and curved faces”, Proc. of Computer Animation 2001, pp. 220-227 (2001).
8) D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. “Elastically deformable models”, Computer Graphics (Proceedings of SIGGRAPH 87), 21(4):205-214 (1987).
9) A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. “Curved PN triangles”, Symposium on Interactive 3D Graphics 2001, pp. 159-166 (2001).
10) P. Volino, M. Courshesnes, and N. M. Thalmann. “Versatile and efficient techniques for simulating cloth and other deformable objects”, Proceedings of SIGGRAPH 95, pp. 137-144, (1995).
11) A. Witkin and D. Baraff. “Differential equation basics”, SIGGRAPH Course Note: Physically-based Modeling. ACM SIGGRAPH (1994).