Parallel simulation of inextensible cloth

Jan Bender and Daniel Bayer

Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe, Germany

Abstract

This paper presents an efficient simulation method for parallel cloth simulation. The presented method uses an impulse-based approach for the simulation. Cloth simulation has many application areas like computer animation, computer games or virtual reality. Simulation methods often make the assumption that cloth is an elastic material. In this way the simulation can be performed very efficiently by using spring forces. These methods disregard the fact that many textiles cannot be stretched significantly. The simulation of inextensible textiles with methods based on spring forces leads to stiff differential equations which cause a loss of performance. In contrast to that, in this paper a method is presented that simulates cloth by using impulses. The mesh of a cloth model is subdivided into strips of constraints. The impulses for each strip can be computed in linear time. The strips that have no common particle are independent from each other and can be solved in parallel. The impulse-based method allows the realistic simulation of inextensible textiles in real-time.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

The dynamic simulation of cloth and fabrics is an important area of research in computer graphics. Simulation methods often treat cloth as an elastic material due to performance reasons. Using this assumption the simulation can be performed efficiently using mass-spring systems. Many textiles are not very elastic and do not stretch significantly under their own weight. Methods based on spring forces need very stiff springs for a realistic simulation regarding this fact. This leads to stiff differential equations which decrease the stability of the simulation [HES03]. These equations can only be solved by special integration methods or by reducing the time step size. In both cases the performance is also decreased significantly. Therefore, these methods are not suitable for the simulation of inextensible cloth.

This paper presents a new constraint-based approach for this problem. In the simulation cloth is handled as a mesh of particles linked by distance constraints. These constraints are satisfied by the computation of impulses. In contrast to methods based on spring forces, the presented impulse-based method can guarantee a predefined accuracy. This is a necessary property for a realistic simulation of inextensible textiles. In order to increase the performance of the simulation first the cloth model is subdivided in acyclic parts. Then the constraint dependencies of each part are described by a system of linear equations. If the structure of the model is taken into account this system can be transformed in an always sparse system. The sparse system can be solved in $O(n)$ time and stored in $O(n)$ space which is the optimal complexity for this problem. In this way the required impulses for each acyclic part can be determined very efficiently. In order to resolve the dependencies between the single parts an iterative approach is used.

The acyclic parts can be divided in two groups where all parts of one group do not depend on another part of the same group. Therefore, the computation of the impulses for all parts of a group can be performed in parallel. This is an important feature, since multi-core systems are already very popular.

The impulse-based method presented in this paper allows the real-time simulation of complex cloth models with a high degree of accuracy. This is shown in the following sections.

2. Related work

Since in [TPBF87] the first general physical model for simulating two and three-dimensional deformable models was presented, the research in this area has undergone a long history. [TPBF87] used a direct matrix solver to compute the (semi-)implicit time integration. Because this technique was

not suitable for larger models, further research generally relied on the use of faster explicit integration, like Euler or Runge-Kutta methods (e.g. [BHW94, CYTT92]). The used models are based on a mesh of particles which are linked by springs and dampers.

The major drawback of using an explicit integration scheme is, that large spring constants lead to "stiff" systems of equations (see [HES03]). These large constants are needed to limit the strain of the cloth which is naturally hard to stretch and easy to bend and shear. The border case, that the cloth is completely inextensible, leads to infinite spring constants and thereby infinite spring forces. For that reason a very small step size is required in order to get a realistic simulation.

To restrict the strain to a certain limit, [Pro95] used an iterative post-processing algorithm which moves particles back into the right position if their distance exceeds 10% of the original edge's length. But these displacements might cause self-intersections of the cloth. In order to avoid this [BFA02] used a slightly different post-processing method which is based on velocities to compute collisions, friction and contacts.

Motivated by the problems of the explicit integration the (semi-)implicit integration was rediscovered for the use of cloth simulation. At first [BW98] presented such a technique, which was well analysed by various groups [HE01, VT00, VMT01]. The result was, that the use of an implicit method increases the stability and thereby larger time steps can be performed. On the other hand, a generally, non-linear system of ODEs has to be solved at each time step.

To address the performance problem mixed implicit and explicit methods (so called: IMEX-methods, e.g. [BMF03, EEH00]) were introduced. The basic idea behind this approach is to split the ODEs in stiff and a non-stiff parts. The stiff parts are then solved using a computational expensive implicit method and the non-stiff ones using a fast explicit method.

In [HCJ*05] for example constraints are only used if the strain exceeds 10% or to prevent penetrations. In [GHF*07] constraints are used to restrict the strain using an iterative fast projection method. The implementation is described as a velocity filter to incorporate with existing simulation code or other velocity filters (like collision response).

Another mixed scheme is proposed in [MHTG05, MHHR07]. This so called position based dynamics method takes explicit Euler integration steps for a preview of the new positions. In an iterative post-processing step these positions are altered until they satisfy the constraints.

A drawback of multiple velocity filters, or post and pre-processing methods, is the independence of the different passes. The next pass may violate the already enforced constraints of the prior one. Therefore, ideal constraint enforcement can only be achieved if the passes are combined.

Other works are based on simplifications to circumvent the performance problem by the cost of quality. For example [KCC*00] and [MDDB01] are using a filter matrix to simulate several hundred particles. This work was extended in [KC02] by using a sparse mesh for global movement and a fine mesh for details. Some other works using multi resolution or adaptive techniques are [DDCB01, GKS02, ST08].

[HB00, Bri03] and [MTV05] give a comprehensive survey on cloth animations. Current research problems in clothing simulation are summarized in [CK05].

3. Cloth simulation

In the dynamic simulation cloth is represented by a mesh of particles. On each edge of the mesh a distance constraint is defined for the corresponding particles. The following sections describe first the simulation of an unconstrained particle. Then a distance constraint is introduced that can be satisfied by computing a single impulse. The mesh of the simulated cloth is subdivided into acyclic parts. It is shown that these parts can be simulated in linear time. At the end of the section a parallel method for the simulation of a whole mesh of particles, linked by distance constraints, is presented.

3.1. Particle simulation

A particle is a body consisting of a single point with no dimensions. Although a particle has no volume it has a mass m. The state of a particle during the simulation is defined by its position $c(t)$ and its velocity $v(t)$ at time t. The motion of an unconstrained particle only depends on the external forces, like gravity, acting on the particle. A simulation step is performed by integrating its state over time regarding these forces. In the following, it is assumed that the sum of all external forces F_{ext}, acting on a particle, is constant during a simulation step. In this case its velocity and its position can be determined by these equations:

$$v(t_0+h) = v(t_0) + \int_0^h \frac{F_{\text{ext}}}{m} dt = v(t_0) + \frac{F_{\text{ext}}}{m} h \quad (1)$$

$$\begin{aligned} c(t_0+h) &= c(t_0) + \int_0^h v(t_0) + \frac{F_{\text{ext}}}{m} t \; dt \quad (2) \\ &= c(t_0) + v(t_0)h + \frac{F_{\text{ext}}}{2m} h^2 \end{aligned}$$

where h is the time step size. If the external forces are not constant during the time step, numerical integration methods must be used. In addition to dynamic particles, the simulation also supports static particles which have no velocity and a fixed position. These particles can be used to fix a cloth model to certain points.

3.2. Distance constraints

In a mesh of a cloth model a distance constraint is defined for each edge. This constraint ensures that the distance of the

two particles at the end points of the edge stays constant over time. There exist different approaches to satisfy such constraints. A simple approach is to introduce a damped spring for each constraint. The spring has the length of the corresponding edge. When the constraint is not satisfied during the simulation, a spring force is acting on the particles that reduces the occurring error. Elastic cloth can be simulated using this method but it is not suitable for the simulation of inextensible textiles, since the constraint is not exactly satisfied. In the following, it is described how a distance constraint is solved exactly by applying impulses to the corresponding particles.

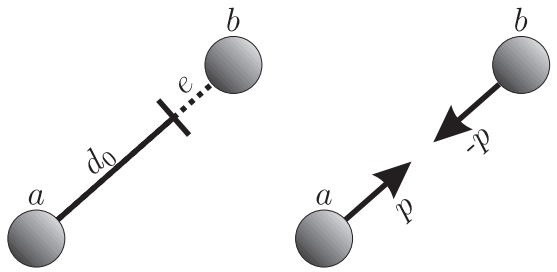
A distance constraint defines a constraint for the positions of two particles a and b and for their velocities. The distance of the particles at time t is determined by

$$d(t) = |c_a(t) - c_b(t)|.$$

In the simulation this distance must stay constant over time. This is realised by the following position constraint:

$$d(t) - d_0 = 0$$

where d_0 is the distance at the beginning of the simulation. In order to satisfy the position constraint an impulse is computed and applied. This impulse is determined by using a preview of the constraint state. The distance of the particles $d(t+h)$ after a simulation step of size h is computed by integrating the positions of the corresponding particles using equation 2. The difference $e_{\text{pos}} = d(t+h) - d_0$ describes exactly the error that would occur if the simulation step was performed without regarding the constraint (see figure 1(a)). In order to prevent this error an impulse p is applied at the beginning of the simulation step. The same impulse must be applied in opposite directions to the particles in order to satisfy the conservation of momentum (see figure 1(b)). Since



(a) Preview (b) Correction impulses

Figure 1: *Correction of a position constraint*

the relative motion of the particles is linear, the impulse must change their relative velocity by e_{pos}/h to correct the error e_{pos} in a time step of size h. Therefore, the required impulse is determined by solving the equation

$$\Delta v_a(p) - \Delta v_b(-p) = (c_b(t) - c_a(t))\frac{e_{\text{pos}}}{h}$$

where $\Delta v_a(p)$ is the velocity change of particle a, when the impulse p is applied. In order to support static and dynamic particles the velocity change is computed by

$$\Delta v_a(p) = k_a\, p$$

where k_a is defined as follows

$$k_a = \begin{cases} \frac{1}{m_a} & \text{if particle } a \text{ is dynamic} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The resulting equation for the correction impulse is

$$(k_a + k_b)\, p = (c_b(t) - c_a(t))\frac{e_{\text{pos}}}{h}.$$

This equation has an unique solution if the particles have different positions and at least one of the particles is dynamic. If the distance constraint is satisfied at the beginning of the step, the first condition is met. The impulse p must be applied in positive direction to particle a and in negative direction to particle b at the beginning of the simulation step. The resulting velocity change will satisfy the position constraint at time $t+h$.

The distance constraint also defines a constraint for the velocities of the particles:

$$(v_b(t) - v_a(t))\,(c_b(t) - c_a(t)) = 0.$$

This constraint prevents the particles from having a different velocity in direction of the constraint. After a simulation step, the constraint is generally not satisfied due to the impulses applied for the correction of the position constraint. In order to satisfy the velocity constraint another impulse is computed. This time no preview is required, since an impulse changes the velocity of a particle immediately. The equation for the impulse is

$$(k_a + k_b)\, p = (c_b(t) - c_a(t))\, e_{\text{vel}}.$$

where the error e_{vel} is determined by the difference of the particle velocities

$$e_{\text{vel}} = (v_b(t) - v_a(t))\,(c_b(t) - c_a(t)).$$

The correction of the velocity constraint provides a higher degree of accuracy but it is not essential for a stable simulation of a distance constraint. Therefore, the velocity correction can even be skipped completely to achieve a better performance.

3.3. Linear-time simulation of acyclic models

Any dynamic model consisting of particles that are linked by distance constraints can be subdivided into parts with an acyclic constraint structure. In this section an impulse-based method is introduced that allows the simulation of such acyclic parts in linear time using linear space. First, it is shown how the impulses for multiple distance constraints can be determined at once using a system of linear equations. Then, this system is transformed into an equivalent system

which is always sparse and which can be solved in linear time.

In the previous section a distance constraint has been introduced. The impulses for this constraint can be determined directly by solving simple equations. If a particle is part of multiple distance constraints, then the impulse of one constraint influences the correction of the other constraints. This means that constraints with a common particle depend on each other. There are different ways to solve these dependencies.

A simple approach is to correct the constraints in an iterative process. This process converges and the result is physically correct (the proof is given in [SBP05]). This iterative method has the advantage that even models with cycles in the constraint structures can be simulated. But the problem of this approach is, that the process converges slowly if the simulated model has many dependencies. Hence, this approach is not suitable for the simulation of a mesh model.

A mesh model can be subdivided into acyclic parts. The dependencies of each part can be described by a system of linear equations:

$$A\,p = \Delta v. \tag{4}$$

The matrix $A \in \mathbb{R}^{n_c \times n_c}$ represents the constraint structure of the model where n_c is the number of distance constraints in the mesh. For each constraint the vector p contains the magnitude of the corresponding correction impulse that should be determined. The vector Δv on the right side contains the errors of all joints that must be corrected. In order to use the same system of linear equations for the position and the velocity correction, the entries of the vector Δv are defined as follows:

$$\Delta v_i = \begin{cases} \frac{e_{\text{pos},i}}{h} & \text{in the case of position correction} \\ e_{\text{vel},i} & \text{in the case of velocity correction.} \end{cases}$$

The number of constraints n_c in the simulated model determines the size of the system of linear equations. Each row and each column of the matrix A correspond to a certain constraint. The value $A_{i,j}$ of the matrix describes how the constraint i depends on the constraint j. Therefore, it is not zero if and only if the corresponding constraints have a common particle. When computing the values of the matrix, it has to be considered, whether this common particle is the first or the second particle of the corresponding constraints. To take this into account, the following matrix is defined:

$$B_{i,j} = \begin{cases} k_{i_1} & \text{if } i_1 = j_1 \wedge i_2 \neq j_2 \\ k_{i_2} & \text{if } i_2 = j_2 \wedge i_1 \neq j_1 \\ -k_{i_1} & \text{if } i_1 = j_2 \wedge i_2 \neq j_1 \\ -k_{i_2} & \text{if } i_2 = j_1 \wedge i_1 \neq j_2 \\ k_{i_1} + k_{i_2} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where i_1 is the index of the first particle of joint i and the value k_i is determined by equation 3.

If two constraints depend on each other, the correction impulses of both constraints influence the velocity of their common body. Hence, when computing the impulses for a constraint, the side effects of all dependencies must be regarded. Therefore, the impulses of the dependencies are projected into the space of the actual constraint. The required projection matrix is defined as follows:

$$P_i = \left(c_{i_2}(t) - c_{i_1}(t)\right)^T \in \mathbb{R}^{1\times 3}.$$

The matrix A of the system of linear equations is determined by projecting the matrices $B_{i,j}$ using the projection matrices $P_i(t)$:

$$\begin{pmatrix} P_1 B_{1,1} P_1^T & \dots & P_1 B_{1,n} P_n^T \\ \vdots & \ddots & \vdots \\ P_n B_{n,1} P_1^T & \dots & P_n B_{n,n} P_n^T \end{pmatrix} \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} \Delta v_1 \\ \vdots \\ \Delta v_n \end{pmatrix}$$

where p_i is the magnitude of the i-th correction impulse. The corresponding three-dimensional impulses are computed by

$$p_i' = p_i P_i^T.$$

The presented system of linear equations takes all dependencies of the model into account. Hence, all correction impulses can be determined in a single step. The system can be solved by using a LU factorization for example which has a time complexity of $O(n^3)$. Since the system is often sparse, even special solvers like PARDISO [SG04] can be used to improve the performance of the simulation. Regarding the fact that the simulated parts have an acyclic constraint structure, the system can even be solved in linear time which is shown in the following.

First, the system of linear equations for the correction impulses must be transformed in the following form:

$$CM^{-1}C^T x = \Delta v, \tag{6}$$

where C is a block matrix that represents the distance constraints and M is the mass matrix of all particles that are part of a constraint. M is a block matrix which contains the mass matrices of all particles on the diagonal. The mass matrix M_i of a single particle with index i is defined by:

$$M_i = \begin{pmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{pmatrix}. \tag{7}$$

Each row in the constraint matrix C represents a distance constraint in the model and each column block a particle. In a system with n_c constraints and n_p particles the matrix C has the dimension $n_c \times 3n_p$. A block $C_{i,j} \in \mathbb{R}^{1\times 3}$ of the matrix is not zero if and only if j is the index of a dynamic particle which is part of the constraint with index i. If the

system of linear equations for the impulses (see equation 4) can be transformed in the form of equation 6, then

$$\underbrace{\begin{pmatrix} M & -C^T \\ -C & 0 \end{pmatrix}}_{H} \begin{pmatrix} y \\ x \end{pmatrix} = \underbrace{\begin{pmatrix} 0 \\ -\Delta v \end{pmatrix}}_{b}$$

is an equivalent system. This system is larger but has the advantage that the matrix H is always sparse. To solve this new system in linear time, first the matrix H must be rearranged. Therefore, a graph is required which represents the constraint structure. An example for such a graph is shown in figure 2. Each particle p_i is represented by a box in the graph

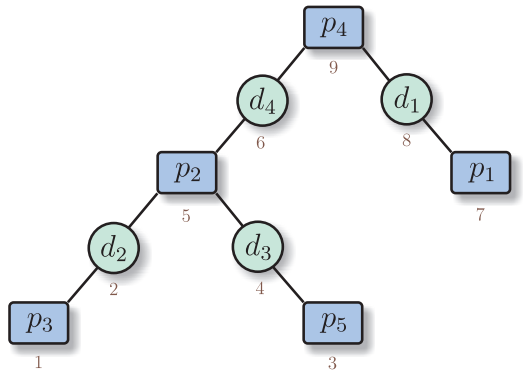


Figure 2: *The constraint structure graph of a simple model. The particles p_i are linked by the distance constraint d_j.*

and each distance constraint d_j by a circle. The new order of the rows and columns of matrix H are found by a depth-first search in this graph. The numbers in figure 2 under the nodes show the resulting row and column indices of the corresponding particles and constraints respectively. Hence, the rows and columns of particles and constraints alternate in the resulting matrix H. After rearranging the matrix the row and column index of each simulation object is greater than the indices of its children. The factorization of the matrix is performed by a decomposition $H = LDL^T$ where L is a lower-triangular matrix whose diagonal values are 1 and D is a diagonal matrix. The new order of H has the effect that the decomposition does not introduce new nonzero elements to the matrix. David Baraff shows in [Bar96] that a matrix with the described property can be solved in $O(n)$ time and stored in $O(n)$ space by using algorithm 1 for the factorization of H. This factorization requires linear time.

Algorithm 1: Factorization

for $i \leftarrow 1$ **to** n **do**
  foreach $j \in children(i)$ **do**
    $H_{i,i} = H_{i,i} - H_{j,i}^T H_{j,j} H_{j,i}$
  if $i \neq n$ **then**
    $H_{i,\text{parent}(i)} = H_{i,i}^{-1} H_{i,\text{parent}(i)}$

The system of linear equations is then solved by using algorithm 2 which also has a time complexity of $O(n)$.

Algorithm 2: Solution

for $i \leftarrow 1$ **to** n **do**
  $x_i = b_i$
  foreach $j \in children(i)$ **do**
    $x_i = x_i - H_{i,j}^T x_j$
for $i \leftarrow n$ **to** 1 **do**
  $x_i = H_{i,i}^{-1} x_i$
  if $i \neq n$ **then**
    $x_i = x_i - H_{i,\text{parent}(i)} x_{\text{parent}(i)}$

After solving the system of linear equations the vector x contains the magnitudes of all correction impulses. The value x_i must be multiplied with the direction of the constraint i at the actual point of time to get the three-dimensional impulse:

$$p_i = x_i P_i^T .$$

The resulting impulses solve all constraints at once.

The linear-time algorithm presented above can only be used if a decomposition $A = CM^{-1}C^T$ of the matrix A exists. The mass matrix M is already known. Since it is a diagonal matrix, its inverse M^{-1} is also a diagonal matrix. The matrix M contains a block M_i (see equation 7) for each dynamic particle on the diagonal. Hence, the diagonal blocks of the inverse of M are computed as follows:

$$M_i^{-1} = \begin{pmatrix} \frac{1}{m_i} & 0 & 0 \\ 0 & \frac{1}{m_i} & 0 \\ 0 & 0 & \frac{1}{m_i} \end{pmatrix} = \begin{pmatrix} k_i & 0 & 0 \\ 0 & k_i & 0 \\ 0 & 0 & k_i \end{pmatrix} .$$

Therefore, only the block matrix C has to be determined.

A block of matrix A for two constraints with a common particle is computed by the equation

$$A_{i,j} = P_i B_{i,j} P_j^T .$$

This block describes how the velocity of constraint i changes, when an impulse is applied to the common particle l for the correction of constraint j. For the decomposition of matrix A the block $A_{i,j}$ must be described by the constraint blocks $C_{i,l}$ and $C_{j,l}$ and by the mass matrix M_l of particle l in the following form:

$$P_i B_{i,j} P_j^T = C_{i,l} M_l^{-1} C_{j,l}^T . \tag{8}$$

Both sides of the equation look already very similar. The value $B_{i,j}$ describes the inverse mass of the common body l of constraint i and j just like the matrix M_l^{-1}. The problem is that $B_{i,j}$ is computed by a case differentiation (see equation 5) which determines the sign of the value. Since the matrix M_l^{-1} is well-defined, this case differentiation must be

performed for the constraint blocks. This is done in the following form:

$$C_{i,l} = \begin{cases} \tilde{C}_{i,l} & \text{if } l = i_1 \\ -\tilde{C}_{i,l} & \text{if } l = i_2 \\ 0 & \text{otherwise} \end{cases}$$

where i_1 and i_2 are the first and second particle of constraint i respectively. The block $\tilde{C}_{i,l}$ must be equal to the projection matrix of the constraint i in order to satisfy equation 8:

$$\tilde{C}_{i,l} = P_i = \left(c_{i_2}(t) - c_{i_1}(t)\right)^T .$$

So, the system of linear equations of the acyclic model can be transformed in the form of equation 6 which is the condition for the linear time method.

The equations above demonstrate how the decomposition of the matrix A is determined for an arbitrary acyclic model. Since the decomposition can always be found, the dynamic simulation of an acyclic model of particles linked by distance constraints can always be performed in linear time and only linear space is required for the computation.

3.4. Parallel mesh simulation

In the previous section a method has been introduced for the linear-time simulation of acyclic models. Since in general a cloth model has no acyclic structure, the mesh must be subdivided in smaller acyclic parts. Two of these parts depend on each other if they have a common particle. These dependencies can even have a cyclic structure which must be resolved. The parts that have no common particle are independent from each other and therefore can be solved in parallel.

At the beginning of section 3.3 it was mentioned that the distance constraints of a model can also be solved in an iterative process. The advantage of this approach is that even models with cycles can be simulated. But for complex models with many dependencies the iterative process converges slowly.

The combination of the linear time method and the iterative approach solves the problem with the cycles in the constraint structure and still shows a good performance. The acyclic parts are divided into groups of independent parts. This means that each part of a group has no dependency with another part of the same group. In general only two different groups are required for a cloth model. All parts of a group can be processed in parallel, since they are independent from each other.

In an iteration step the groups are processed one after another. The impulses for each part of a group are determined using the linear-time method presented in this paper. Since the matrix A of the system of linear equations is constant at time t, the factorization for each part has to be performed only once per simulation step. The matrix for the velocity constraints equals the one for the position constraints of the same point of time. This means that the factorization of the velocity correction can even be reused for the position correction of the next simulation step.

The factorization of the matrices is the most time-consuming part of the impulse computation. Since this has to be done only once per simulation step, it does not cost much performance to solve the systems multiple times in the iteration process. In the same process collision and contact handling with friction can be performed. In order to support collisions, the method described in [BS06] was integrated in the simulation.

4. Results

In this section results concerning the performance of the impulse-based method are presented. All simulations in this section were performed on a PC with a 2.4 GHz Intel Core 2 Quad processor. The cloth model used for the performance tests consists of a regular grid of particles. This grid is subdivided in horizontal and vertical acyclic strips (see figure 3). Since they are acyclic, the linear-time algorithm of section 3.3 can be used for the simulation of the strips. In the group of all horizontal strips and in the group of all vertical ones, there are no direct dependencies. Therefore, all strips in each group can be solved in parallel.

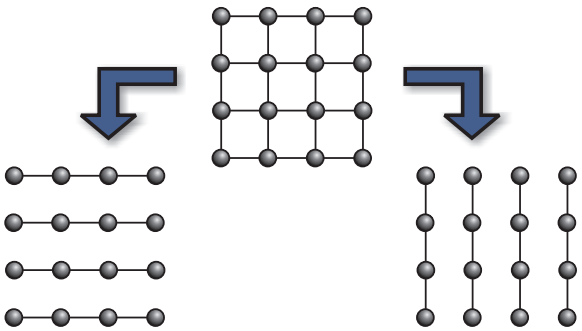
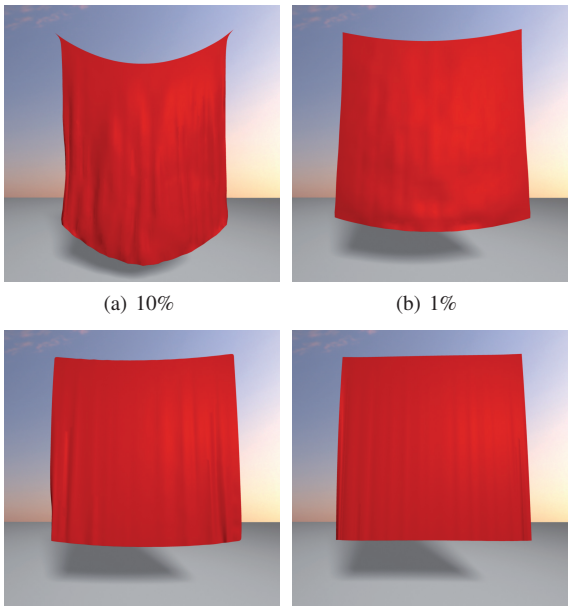


Figure 3: *Example of a regular grid that is subdivided in eight strips*

The tolerance value defines the allowed strain of the textile. Figure 4 shows a cloth model with a 41×41 grid of particles that are linked by 3280 distance constraints. The cloth was pinned at two corners in order to study the relaxation due to gravity. The tolerance was set to values between 0.1 and 0.0001 to simulate an allowed strain between 10% and 0.01%. Even smaller tolerance values can be used for the simulation with the introduced method. In the following, the model of figure 4 is used to measure the performance of the method.

Cloth models with different grid sizes were simulated in order to measure the performance of the simulation method.



(a) 10% (b) 1%

(c) 0.1% (d) 0.01%

Figure 4: *A cloth model was simulated with different tolerance parameters. The tolerance value was chosen so that the allowed strain was between* 10% *and* 0.01%.

Grids of between 10×10 and 50×50 particles were used. The smallest model had 180 distance constraints whereas the largest model had 4900 constraints. The performance does not only depend on the complexity of the mesh, it also depends on the allowed strain. Therefore, the models were simulated with different tolerance values for a performance analysis. The time step size of the simulation was set to $h = \frac{1}{30}$ s in order to get 30 frames per second. At the beginning of the simulation, the cloth was parallel to the floor and some impulses were applied to randomly chosen particles in order to measure the performance under realistic conditions. After 500 simulation steps, the average computation time per frame was determined. The simulation was performed using all four cores of the CPU. Figure 5 shows these average computation times for all models. The horizontal line in the figure marks the value of $\frac{1}{30}$ s. If the simulation steps are performed faster than this time, then the simulation runs faster than real-time.

The 30×30 mesh could even be simulated faster than real-time, when using an allowed strain of 0.1% and 1% percent. The smallest mesh was simulated about six times faster than real-time, when using the smallest tolerance value of 0.0001. The model consisting of 50×50 particles linked by 4900 constraints was not even six times slower than real-time using an allowed strain of 0.01%. The use of four cores instead of one caused a speed-up of a factor of about 1.9 on the used system.

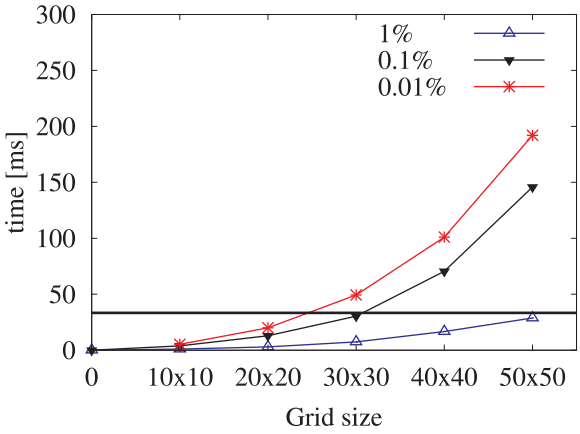


Figure 5: *Average computation times per simulation step for different cloth models simulated with a allowed strain between* 10% *and* 0.01%

If only plausible results are demanded, the simulation can be accelerated by stopping the iterative process after a maximum number of iterations. In this case, the tolerance values are not reached but the simulation is still stable and the results are visually plausible. In a simulation with a maximum of five iterations, the model with a 30×30 mesh and an allowed strain of 0.1% was simulated 3.7 times faster than real-time whereas the 50×50 mesh was simulated in real-time.

Collision and contact handling with friction can be integrated in the iteration process. This is demonstrated in figure 6. The model in the figure was simulated by using the

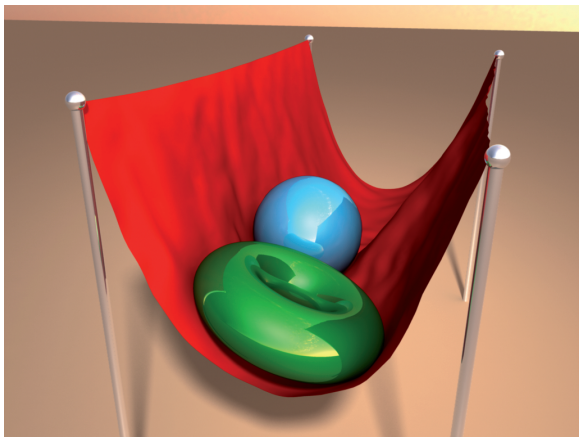


Figure 6: *Example for a cloth simulation with collision handling*

method of Bender et al. [BS06] for resolving the collisions and contacts with friction.

5. Conclusion

An impulse-based method for the simulation of cloth is presented in this paper. In contrast to methods based on mass-spring systems, the impulse-based approach allows the efficient simulation of inextensible textiles. Cloth is simulated by using a mesh of particles linked by distance constraints. These constraints are solved by the computation of impulses. An impulse is determined by using a preview of the state of the corresponding constraint. In this way a constraint is solved exactly in a single step.

If a particle is part of multiple constraints, the constraints depend on each other. The dependencies for an acyclic model are resolved by using a system of linear equations which describes the constraint structure. This system can be solved in $O(n)$ time and stored in $O(n)$ space which is the optimal complexity. In general, a cloth model contains cycles in its constraint structure. To solve this problem the model is subdivided in acyclic parts. Each part is then simulated in linear time. To solve the cyclic dependencies between the single parts an iterative method is used. Since parts with no common particle do not directly dependent on each other, in an iteration step, their correction impulses can be computed in parallel.

The presented approach allows the real-time simulation of complex cloth models. The method guarantees a predefined accuracy which is defined by the used tolerance value. Since a distance constraint is directly satisfied by using a preview, even destroyed models can be repaired. Therefore, an early result can be obtained by interrupting the iterative process of a simulation step without the loss of stability. This provides another way to increase the performance. Collision and contact handling with friction can also be integrated in the simulation process.

References

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 137–146.

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 594–603.

[BHW94] BREEN D. E., HOUSE D. H., WOZNY M. J.: Predicting the drape of woven cloth using interacting particles. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM, pp. 365–372.

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles, 2003.

[Bri03] BRIDSON R. E.: *Computational aspects of dynamic surfaces*. PhD thesis, Stanford, CA, USA, 2003. Adviser-Ronald Fedkiw.

[BS06] BENDER J., SCHMITT A.: Constraint-based collision and contact handling using impulses. In *Proceedings of the 19th international conference on computer animation and social agents* (Geneva (Switzerland), July 2006), pp. 3–11.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. *Computer Graphics 32*, Annual Conference Series (1998), 43–54.

[CK05] CHOI K.-J., KO H.-S.: Research problems in clothing simulation. *Computer-Aided Design 37*, 6 (2005), 585–592.

[CYTT92] CARIGNAN M., YANG Y., THALMANN N. M., THALMANN D.: Dressing animated synthetic actors with complex deformable clothes. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1992), ACM, pp. 99–104.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 31–36.

[EEH00] EBERHARDT B., ETZMUSS O., HAUTH M.: Implicit-explicit schemes for fast animation with particle systems. In *Proc. Eurographics Workshop Computer Animation and Simulation* (2000), pp. 137–154.

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Transactions on Graphics 26*, 3 (2007), 49.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: a simple framework for adaptive simulation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 281–290.

[HB00] HOUSE D. H., BREEN D. E. (Eds.): *Cloth modeling and animation*. A. K. Peters, Ltd., Natick, MA, USA, 2000.

[HCJ*05] HONG M., CHOI M.-H., JUNG S., WELCH S., TRAPP J.: Effective constrained dynamic simulation using implicit constraint enforcement. In *International Conference on Robotics and Automation* (Apr 2005).

[HE01] HAUTH M., ETZMUSS O.: A high performance solver for the animation of deformable objects using advanced numerical methods. In *EG 2001 Proceedings*, Chalmers A., Rhyne T.-M., (Eds.), vol. 20(3). Blackwell Publishing, 2001, pp. 319–328.

[HES03] HAUTH M., ETZMUSS O., STRASSER W.:

Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7-8 (2003), 581–600.

[KC02] KANG Y.-M., CHO H.-G.: Bilayered approximate integration for rapid and plausible animation of virtual cloth with realistic wrinkles. In *CA '02: Proceedings of the Computer Animation* (Washington, DC, USA, 2002), IEEE Computer Society, p. 203.

[KCC*00] KANG Y., CHOI J., CHO H., LEE D., PARK C.: Real-time animation technique for flexible and thin objects. In *Proceedings of WSCG* (2000), pp. 322–329.

[MDDB01] MEYER M., DEBUNNE G., DESBRUN M., BARR A. H.: Interactive animation of cloth-like objects in virtual reality. *The Journal of Visualization and Computer Animation 12*, 1 (2001), 1–12.

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (2007), 109–118.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 471–478.

[MTV05] MAGNENAT-THALMANN N., VOLINO P.: From early draping to haute couture models: 20 years of research. *The Visual Computer 21*, 8-10 (2005), 506–519.

[Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95* (1995), Davis W. A., Prusinkiewicz P., (Eds.), Canadian Human-Computer Communications Society, pp. 147–154.

[SBP05] SCHMITT A., BENDER J., PRAUTZSCH H.: *On the Convergence and Correctness of Impulse-Based Dynamic Simulation*. Internal Report 17, Institut für Betriebs- und Dialogsysteme, 2005.

[SG04] SCHENK O., GÄRTNER K.: Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems 20*, 3 (2004), 475–487.

[ST08] SPILLMANN J., TESCHNER M.: An adaptive contact model for the robust simulation of knots. *Computer Graphics Forum 27*, 2 (apr 2008), 497–506.

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM, pp. 205–214.

[VMT01] VOLINO P., MAGNENAT-THALMANN N.: Comparing efficiency of integration methods for cloth simulation. In *CGI '01: Computer Graphics International 2001* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 265–274.

[VT00] VOLINO P., THALMANN N. M.: Implementing fast cloth simulation with collision response. In *CGI '00: Proceedings of the International Conference on Computer Graphics* (Washington, DC, USA, 2000), IEEE Computer Society, p. 257.