# Simulating, animating and rendering clothes

Guenter Wallner

## Abstract

This paper presents a retrospective of past work in the field of cloth simulation and rendering. Cloth animation is crucial for the rendering of a scene involving virtual actors. Over the years a broad range of methods have been proposed to animate cloth in a fast and stable way. Furthermore, collisions of the cloth, either with itself or other objects in the scene are a challenging area. To date a variety of work has been published about how to efficiently detect and resolve collisions. The internal microstructure of cloth makes it difficult to render garments in a fast and sophisticated way. Special rendering techniques to overcome the problems will therefore be reviewed. The availability of programmable GPUs has resulted in a great body of work about real-time simulation of cloth on graphics hardware.

**Keywords:** cloth rendering, collision detection, particle system, real time, BRDF

## 1 Introduction

Cloth is a major part of our daily lives and comes in a myriad of types. If we want to animate humans the simulation of cloth can not be dismissed. Cloth modeling research started in the 1930s in the textile engineering community. Weil [Weil 1986] was the first who proposed a computer animation model of cloth. While Weil used a geometric approach, Terzopoulos et al. [Terzopoulos et al. 1987] proposed a physically based simulation for fabric behavior. Since then various other physically based techniques have been developed (for example by [Carignan et al. 1992; Breen et al. 1992; Provot 1995; Volino et al. 1995; Volino and Magnenat-Thalmann 1997b]). Until Breen et al. [Breen et al. 1992] most of the efforts to create a model of cloth have used continuum mechanics. However, according to Breen et al. [Breen et al. 1992] these techniques rely on the assumption that the behavior of a small element can be described with the same continuum equations that describe the large scale behavior, which holds not true for cloth. They therefore proposed a particle-based model of cloth. Provot [Provot 1995] described a mass and spring system with a specific spring configuration for cloth modeling. All of these methods have in common that they rely on numerically solving an ordinary differential equation. A major step for cloth simulation was the work by Baraff and Witkin [Baraff and Witkin 1998], who used an implicit integration scheme that circumvents the numerical instabilities exhibited by explicit integration when using large time steps. Most recently Choi [Choi and Ko 2005] published a method which accounts for the so called post-buckling instability which has mostly been ignored due to it's problematic nature.

Collision detection is a major and time consuming problem for a sophisticated simulation of cloth, because in contrast to rigid bodies, collisions do not only occur between different objects but also with itself. Therefore a significant body of work has been published. Some works applied techniques commonly used in computer graphics, like hierarchical bounding volumes (e.g. [Volino and Magnenat-Thalmann 1994; Provot 1997]) or spatial subdivision methods (e.g. [House and Breen 2000; Zhang and Yuen 2000]). Volino and Magnenat-Thalmann as well as Mezger et al. [Volino and Magnenat-Thalmann 2000a; Mezger et al. 2003] used k-DOPs [Klosowski et al. 1998] instead of bounding spheres or axis aligned bounding boxes, which reduces the number of bounding volume checks. Provot [Provot 1997] introduced the idea of a normal cone for detecting self intersections. Although, very fast his method can fail to detect self intersections of concave meshes and therefore was extended by Mezger et al. [Mezger et al. 2003] for non-convex polyhedra. Bridson et al. [Bridson et al. 2002] proposed an approach for handling cloth-cloth collisions which produces very sophisticated results, although not feasible for real-time applications. They also presented a post-processing subdivision scheme to smooth the triangle mesh used for simulation. Baraff et al. [Baraff et al. 2003] presented a history-free cloth collision response algorithm. The dependence on history – to guarantee a collision free state – was one of the major shortcomings of past approaches.

With the advent of programmable GPUs researches are utilizing the parallel processing power to speed up cloth simulation and collision handling. In 2004 Simon Green of NVidia [Green 2004] released a demo about cloth simulation on the GPU which uses a Verlet integration, as proposed by Jakobsen [Jakobsen 2003], but could only handle collision detection with a sphere and self-collisions were not taken into account. Zeller [Zeller 2005; Zeller 2006] presented a GPU implementation of the same method, but with improved collision detection and responsiveness to wind. Contrary to the afore mentioned methods, which handle collision detection in world space, image based methods have been employed [Wong 2004; Wang 2005].

Cloth can be made from a variety of different materials and we can easily recognize cloth such as silk or velvet by their color and reflectivity. Moreover, individual stitches or knits can possibly be resolved from certain viewing directions. Stewart [Stewart 1999] addressed the shading of cloth with a visibility cone. His work was improved in regard to computational speed by Ganster et al. [Ganster et al. 2002]. Daubert et al. [Daubert et al. 2001] proposed a method for rendering cloths by replicating individual weaving or knitting patterns, whereas Xu et al. [Xu et al. 2001] introduced the lumislice for high quality off-line rendering. Bidirectional texture functions (introduced by Dana et al. [Dana et al. 1999]) are used by Sattler et al. [Sattler et al. 2003] to capture and visualize reflection properties of cloth at interactive frame rates.

The remainder of this paper is organized as follows. Section 2 will give a short overview of the more important physical properties of cloth. In Section 3 some of the more common cloth modelling/simulation methods are reviewed. Section 4 will give an overview about collision detection methods used in cloth animation. Recent developments in cloth simulation on GPUs are discussed in Section 5. Rendering of cloth is treated in Section 6.

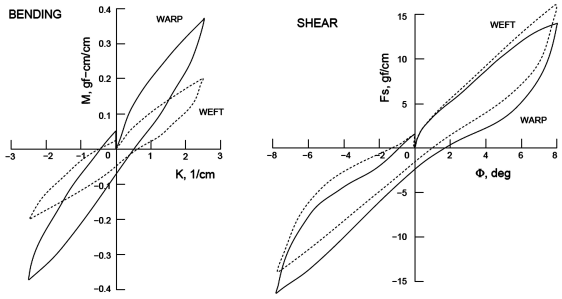


Figure 1: Kawabata bending and shear plots of 100% cotton. It can be observed that the deformation of cloth is different depending whether stress is applied or released (from [Breen et al. 1994]).

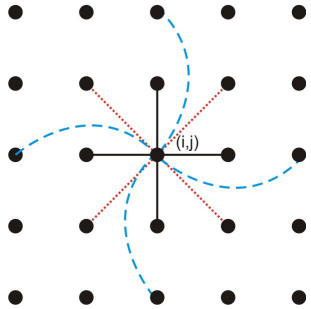


Figure 2: Provot's mass and spring configuration for a mass $(i, j)$. The black solid lines represent structural springs. Red dotted lines depict shear springs and blue lined lines are flexion springs.

## 2 Physical Properties of Cloth

Although researchers in the computer graphics community are generally interested to produce results that look realistic rather than physically accurate solutions, the physical properties of cloth cannot be left completely unconsidered. The textile community proposed several methods over the years to study the mechanics of fabric. Peirce [Peirce 1937] was the pioneer in 1937 when he analyzed the geometrical relationships among yarns at yarn crossings. However, until the late 1960s the different components of cloth mechanical behavior (tensile, bending, and shearing properties) have been studied separately [House and Breen 2000]. Cloth strongly resists stretching motions while being comparatively permissive in allowing bending or shearing motions [Baraff and Witkin 1998]. The Kawabata system ([Kawabata et al. 1973]) proved very useful for computer graphics researches. The system consists of five tests, each measuring a mechanical property of cloth and producing a so called Kawabata plot. Figure 1 shows the Kawabata bending and shearing plots of cotton. To obtain the curves a load is applied to a piece of cloth of standardized dimensions and then released. The procedure is then repeated in the inverse direction. An interesting behavior of cloth can be observed from this plots. The path of deformation when the cloth is stressed is different from the path when the stress is released.

## 3 Methods

According to House and Breen [House and Breen 2000] methods for simulating cloth in three dimensions can roughly be divided in *geometric* and *physically based approaches*. Geometric models derive the cloth motion and deformation from geometrical curves and functions that are parametrized by time [Volino and Magnenat-Thalmann 2000b]. The main concern of geometric models is to provide a fast solution in specific contexts. Physically based methods can further be divided in *mass and spring models*, *elasticity-based models* and *particle models* and will be discussed in more detail in the following subsections.

### 3.1 Mass and Spring Models

Provot [Provot 1995] proposed a mass and spring system with a spring configuration explicitly aimed to cloth simulation. He models cloth as a regular rectangular grid of $m \times n$ virtual masses which are connected by massless springs of natural length non equal to zero. The linkage in between neighbors is achieved by three different kinds of springs. *Structural springs* link mass $[i, j]$ with masses $[i+1, j]$ and $[i, j+1]$ to counteract tension. *Shear springs* connect masses $[i, j]$ and $[i+1, j+1]$, and masses $[i+1, j]$ and $[i, j+1]$. *Flexion springs*, which handle bending, connect mass $[i, j]$ with mass $[i+2, j]$ and $[i, j+2]$ respectively. The configuration can be seen in Figure 2.

As in all discrete models of cloth, the evolution of the system is governed by Newton's second law of motion:

$$\mathbf{F_i} = m_i \mathbf{a_i} \tag{1}$$

where $\mathbf{a_i}$ is the acceleration caused by the force $\mathbf{F_i}$ and $m_i$ is the mass of the point $\mathbf{P_i}$. The force function consists of internal forces and of external forces acting on the cloth. Provot [Provot 1995] modeled the internal forces between two virtual masses $\mathbf{P_i}$ and $\mathbf{P_j}$ as:

$$\mathbf{F}_{int}(P_{i,j}) = k_{i,j}(\|\mathbf{P_i} - \mathbf{P_j}\| - l_{i,j}) \frac{\mathbf{P_i} - \mathbf{P_j}}{\|\mathbf{P_i} - \mathbf{P_j}\|} \tag{2}$$

where $l_{i,j}$ is the rest length and $k_{i,j}$ is the stiffness of the spring linking $\mathbf{P_i}$ and $\mathbf{P_j}$. External forces in Provot's model account for gravity, viscous damping (models energy dissipation due to internal friction) and interaction with an air stream. After the function $\mathbf{F}$ has been determined it can be integrated through time with a numerical integration scheme, like explicit Euler. However, the numerical stability of an explicit integration method depends on an appropriately chosen time step $\Delta t$ (see Appendix A).

In some cases springs are undesirably stretched or compressed by large percentages which lead to the so-called *super-elastic effect*. This can be observed in Figure 3 for the springs directly tied to the fixed corners. The deformation of the most elongated springs exceed 100% (Provot [Provot 1995] referred to it as the deformation rate, although the correct term is strain rate, as noted by Bridson et al. [Bridson et al. 2002]). An edge should not change length by more than 10% in a single time step [Caramana et al. 1998]. Provot [Provot 1995] addressed the problem by using an iterative algorithm to repair such deformed springs after each time step. It must be noted that this approach involves moving nodes, which can lead to self-intersections in the mesh.

Mass-spring systems can be generalized to non-regular triangular meshes. Using a simple mass-spring model as for quads will lead to physical properties that depend on the topology of the mesh, since each edge will produce a force component along its direction, which is usually not the deformation direction. Therefore Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann 1997b] proposed the following model in which the elongation of an edge depends on the interdependence of the displacements generated by

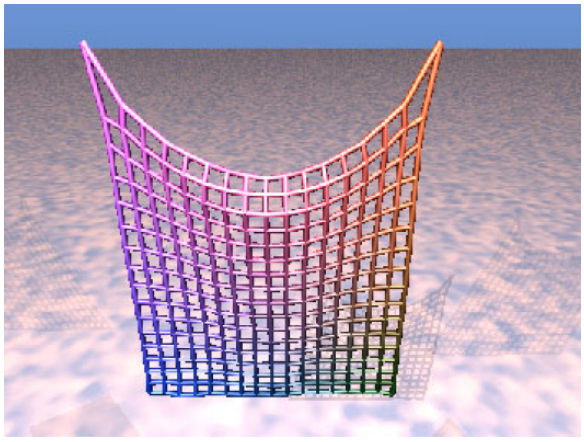

Figure 3: The super-elastic effect can be observed best at the edges connected to the two fixed corners (reprinted from [Provot 1995])

each of them. Assume a triangle with points $\mathbf{P_a}$, $\mathbf{P_b}$ and $\mathbf{P_c}$, then the desired elongation of an edge $\mathbf{L_{ij}}$ corresponds to

$$l_{jk} - L_{jk} = d_{jk} + c_k \frac{M_k^{-1}}{M_k^{-1} + M_i^{-1}} d_{ik} + c_j \frac{M_j^{-1}}{M_j^{-1} + M_i^{-1}} d_{ij} \qquad (3)$$

for all permutations of $i,j,k \in \{a,b,c\}$. $c_i$ is the angle between the two edges at point $\mathbf{P}_i$. Assuming the edge angles do not vary significantly the problem can be linearized. Solving the resulting linear system yields the displacement values $d_{ab}$, $d_{bc}$ and $d_{ca}$.

Mass and spring models are often considered as simple particle models (see Section 3.3), as for example in [Volino and Magnenat-Thalmann 2000b; Volino and Magnenat-Thalmann 1997a]. These models yield very simple computations, but are not very accurate, as an array of springs cannot represent exactly the elastic behavior of a plain elastic surface (as noted in [Volino and Magnenat-Thalmann 1997a]).

## 3.2 Elasticity-Based Models

Opposed to the spring-mass models presented in the previous section where cloth is handled as a discrete surface, cloth is considered as a continuous surface that will have to be discretized before being solved numerically as an ordinary differential equation

$$\ddot{\mathbf{x}} = \mathbf{M^{-1}}(-\frac{\partial E}{\partial \mathbf{x}} + \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}})) \qquad (4)$$

where $\mathbf{x}$ represents the geometric state of the cloth and matrix $\mathbf{M}$ its mass distribution. $E$ is a scalar function describing the cloth's internal energy and $\mathbf{F}$ a function of forces acting on the cloth.

Baraff and Witkin [Baraff and Witkin 1998] describe a simple cloth model based on a continuum representation of cloth, which was significantly faster than previous proposed methods. Their main concern was to overcome numerical computation problems known from other methods. Instead of using an explicit integration scheme they used an implicit integration method to solve Equation 4. Cloth is defined as a triangular mesh of $n$ particles, where $\mathbf{x}_i \in \mathbb{R}^3$ is the position and $\mathbf{f}_i$ is the force of the $i$th particle. Assigning each particle a mass $m_i$ and defining the diagonal mass matrix $\mathbf{M} \in \mathbb{R}^{3n \times 3n}$ by $diag(\mathbf{M}) = (m_1, m_1, m_1, m_2, m_2, m_2, ..., m_n, m_n, m_n)$ Newton's second law of motion can be written as $\ddot{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$ where $\mathbf{f}$ is the net force vector. Baraff and Witkin [Baraff and Witkin 1998] transformed this equation into a first-order differential equation, which is needed for computing the new position and velocity with the implicit Euler method. $\Delta x$ and $\Delta v$ are calculated by

$$\begin{pmatrix} \Delta x \\ \Delta v \end{pmatrix} = \Delta t \begin{pmatrix} \mathbf{v}(t_0) + \Delta \mathbf{v} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}(t_0) + \Delta \mathbf{x}, \mathbf{v}(t_0) + \Delta \mathbf{v}) \end{pmatrix} \qquad (5)$$

with $\Delta \mathbf{x} = \mathbf{x}(t_0 + \Delta t)$ and $\Delta \mathbf{v} = \mathbf{v}(t_0 + \Delta t)$. Using the inverse mass matrix has two advantages, as pointed out by Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann 2000b]. First, multiplications can be computed more efficiently and never introduce singularities. Second, constrained particles can be handled with an inverse mass of zero (a particle of infinite mass has no acceleration). Since this is a non-linear equation $\mathbf{f}$ is approximated by a Taylor series expansion

$$\mathbf{f}(\mathbf{x}(t_0) + \Delta \mathbf{x}, \mathbf{v}(t_0) + \Delta \mathbf{v}) = \mathbf{f}_0 + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Delta \mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \Delta \mathbf{v} \qquad (6)$$

to get a linear system. Because sensible damping functions cannot be derived from energy functions, Baraff et al. [Baraff and Witkin 1998] define internal behavior (stretch, shear and bend) as vector conditions which are zero in their minimal state. Figure 4 shows some results obtained with this method.

Elasticity-based models can be used in a variety of contexts and yield accurate results with complex mechanical properties. Their main drawback are expensive calculations, mainly for solving the huge linear system using an iterative sparse matrix technique[1]. For example, Baraff and Witkin [Baraff and Witkin 1998] solved the matrix with a modified conjugate gradient (CG) method. CG methods exploit sparsity quite easily, since they are based solely on matrix-vector multiplies, and require only rudimentary sparse storage techniques [Baraff and Witkin 1998]. A good introduction on the CG method can be found in [Shewchuk 1994].

## 3.3 Particle Models

Breen et al. [Breen et al. 1992] observed that continuum methods don't produce very sophisticated results when attempting to reproduce folds and buckles. They therefore model cloth as interacting particles representing the crossing of warp and weft threads in a plain weave. The original implementation, which will be reviewed in the remainder of this section, was slow and only concerned with modeling draped configurations. However, Eberhardt et al. [Eberhardt et al. 1996] extended the work to allow a dynamic simulation that shows the effects of air resistance, wind, moving bodies, and surface friction. Beside that, they described a faster way for calculating the particle trajectories.

The structural constraints at thread-level are represented by energy functions. These functions model simple geometric relationships between particles in a local neighborhood and can be summarized for a particle $i$ as

$$E_i = E_{repel_i} + E_{stretch_i} + E_{bend_i} + E_{trellis_i} + E_{grav_i} \qquad (7)$$

[1] Since each particle in a cloth simulation is only connected with a small number of other particles the resulting matrix is sparsely populated.

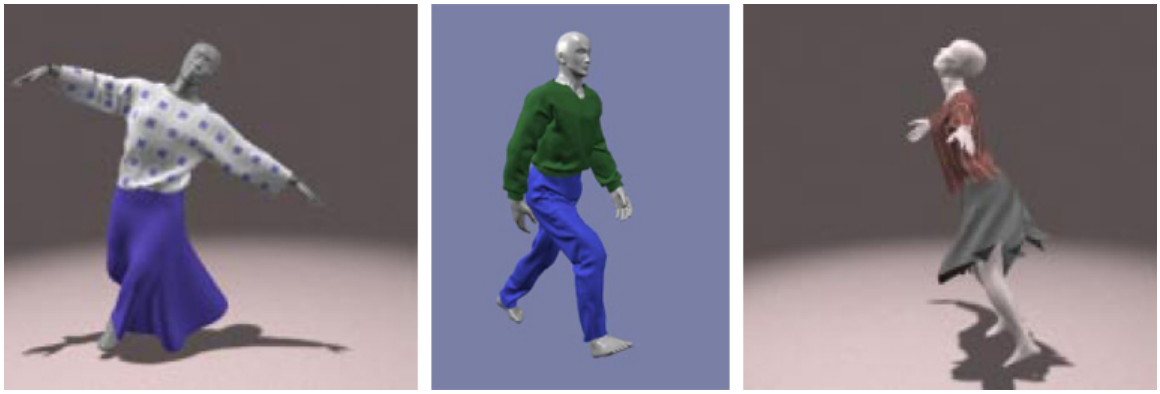

Figure 4: Some sample images from the paper of [Baraff and Witkin 1998].

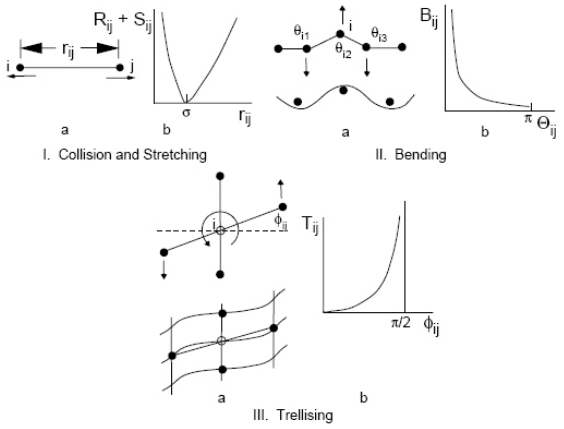


Figure 5: Energy functions for the cloth model of Breen at el. [Breen et al. 1992].

where $E_{repel_i}$ is an artificial energy of repulsion. This energy prevents self intersections by keeping every other particle at a minimum distance. $E_{grav_i}$ accounts for gravity, $E_{strech_i}$ models the tension between particle $i$ and it's four connected neighbors, $E_{bend_i}$ describes out-of-plane bending and $E_{trellis_i}$ is the energy to due the in-plane bending around a thread crossing (see Figure 5). Breen et al. [Breen et al. 1992] report that for the repelling and stretching forces they had good success with

$$R(r_{ij}) = \begin{cases} C_0((\sigma - r_{ij})^5 / r_{ij}) & r_{ij} \leq \sigma \\ 0 & r_{ij} > \sigma \end{cases} \quad (8)$$

$$S(r_{ij}) = \begin{cases} 0 & r_{ij} \leq \sigma \\ C_0((r_{ij} - \sigma)/\sigma)^5 & r_{ij} > \sigma \end{cases} \quad (9)$$

where $r_{ij}$ is the distance between particle $i$ and $j$, $\sigma$ the nominal distance and $C_0$ a scale parameter. $E_{repel_i}$ can then be calculated by summarizing over all particles and $E_{stretch_i}$ by summing over the four connected particles. The bending energy is a function of the angle formed by three particles along a weft or warp line (see Figure 5). The complete bending energy is the sum of the six angles (three in weft and three in warp direction). The effect of trellising can also be seen in Figure 5. To account for the phenomenon, two segments are formed which connect the two pairs of neighboring particles. If cloth has not been deformed yet, the angle between the two segments would be 90°, which is therefore assumed as the equilibrium angle. The trellis angle is the angle formed as one of the line segments moves away from this equilibrium.

Breen et al. [Breen et al. 1992] also implemented a three step process to incorporate the Kawabata plots (see Section 2). First, functions which approximate the Kawabata plots are determined. In the next step, the functions are related to the energy functions discussed above. Finally, the equations are scaled in a way to produce energy values in standard physical units. With their method they were able to reproduce the drape of a specific material accurately, as they show in [Breen et al. 1994].

Jakobsen [Jakobsen 2003] presented a fast method which is particularly well suited for GPU implementations (GPU specific implementation issues will be discussed in Section 5). He uses a Verlet integration scheme which requires no explicitly given velocity. Only the previous and current positions are needed (the velocity is therefore implicitly given). For a current position $x_t$ and a previous position $x_{t-1}$ the integration step can be written as

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \varepsilon(\mathbf{x}_t - \mathbf{x}_{t-1}) + a\Delta t^2 \quad (10)$$

The Verlet integration is not always very accurate but it is quite stable. Similar to other methods, a particle is initialized for every vertex of the mesh and a spring for each edge. Jakobsen [Jakobsen 2003] speaks of sticks, because springs are modelled with infinite stiffness. Which means that springs are not simulated as forces, but as distance constraints. This allowed him to solve the system in a stable and quick way. The stick's "rest length" is in this case simply the initial distance between the two vertices. Several other constraints can be included, e.g to handle collisions. The constraints are solved indirectly by local iteration (relaxation).

## 4 Collision Resolution

Collision Resolution consists of detecting collisions and resolving them properly and poses a major bottleneck in cloth simulation. The main problem lies in the complexity arising from the discretized meshes. Several methods have been proposed to reduce the complexity by using bounding volume hierarchies or spatial subdivision. Additional difficulties arise from the fact that cloth is not a rigid body and therefore self-intersections can occur. Once a collision has been detected, it must be resolved in a physical plausible way. Choi and Ko [Choi and Ko 2003] note, that collision detection

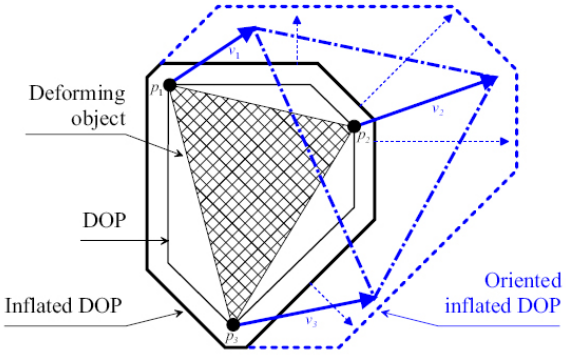


Figure 6: A triangle enclosed by an 8-DOP. The $k$-DOP is inflated along the normalized mean axis to enclose the space traversed by the primitive during a time step. Reprinted from [Mezger et al. 2003].

for cloth should aim to detect more collisions that actually occur, rather than missing a real collision. Because missing a real collision can lead to erroneous results in the remainder of the simulation.

### 4.1 Bounding Volume Hierarchies

If collisions between two animated polygonal meshes have to be detected, bounding techniques based on polygon hierarchies seem to be the best methods, as noted by Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann 2000a]. Because if the surface is moving, only the bounding volumes have to be recalculated and the hierarchy can be kept constant. The performance of hierarchical techniques highly depend on the hierarchical tree, which should fulfill the following criteria [Volino and Magnenat-Thalmann 2000a; Volino and Magnenat-Thalmann 2000b]: First, each node should have a maximum of $O(1)$ children and second, the tree should be balanced (a maximum depth of $O(\log n)$). As bounding volumes mainly Axis Aligned Bounding Boxes (AABBs) have been used (e.g. [Provot 1997; Volino and Magnenat-Thalmann 2000a; Bridson et al. 2002]). However, AABBs do not always enclose the polygons very tight.

Therefore, $k$-DOPs [Klosowski et al. 1998] are used which can be made to approximate the convex hull by increasing $k$ [Teschner et al. 2004]. Following the definition of Funfzig and Fellner [Funfzig and Fellner 2003], a $k$-DOP is defined by a fixed small set of $k$ directions $(\mathbf{n}_1,...,\mathbf{n}_k)$ and a tuple $(d_1,...,d_k) \in \mathbb{R}^k$ of scalars by

$$\bigcap_{i=1}^{k} H_i = \{\mathbf{p} | \mathbf{n}_i \mathbf{p} \leq d_i, i = 1,...,k\} \tag{11}$$

with half-space $H_i$ denoted as

$$H_i := \{\mathbf{p} \in \mathbb{R}^3 | \mathbf{n}_i \mathbf{p} \leq d_i\} \tag{12}$$

If the hyperplanes form $k/2$ parallel lines the intersection test for a polyhedron turns into a simple interval test [Mezger et al. 2003]. Note that an AABB is a special case of a $k$-DOP with $k = 6$. Figure 6 shows an 8-DOP. Such DOPs are generally faster to update for deformable objects than Oriented Bounding Boxes (OBB) [Teschner et al. 2004] and therefore preferred for cloth simulation.

Although large time steps are desirable for fast cloth simulation as described in the previous section, they impose difficulties for collision detection. To provide a stable collision detection, collisions should be detected before they occur rather than being corrected after an invalid state has been reached. Therefore not only colliding faces should be detected, but also proximities between faces. This means that the bounding volume must enclose the space between two consecutive frames which is likely to be traversed and must be enlarged by $\varepsilon_{close}/2$ in each direction. $\varepsilon_{close}$ is the maximum distance of two meshes where proximities have to be detected. To determine the space which will be traversed, the next time step and the velocities of the vertices must be estimated to extrapolate the new vertex positions. Since this would double the cost of updating the leaves of the hierarchy, as noted by Mezger et a. [Mezger et al. 2003], they introduced the orientated $k$-DOP inflation. This methods updates each of the $k/2$ intervals depending on the normalized mean axis $\bar{v}$ and the maximal velocity $a_{max}$ of the velocity cone[2]. The interval limits are increased by the distance

$$dist_i = \varepsilon_{close} + \max(\langle \bar{\mathbf{v}}, \mathbf{n}_i \langle a_{max} \Delta t, 0) \tag{13}$$

where $\Delta t$ is the expected time step. If the velocity cone has no principal direction the inflation $dist = \max(\varepsilon_{close}/2, a_{max}\Delta t)$ is applied. Figure 6 shows the oriented $k$-DOP inflation for a triangle.

The hierarchy of $k$-DOPs can either be build bottom-up or top-down (which is the preferred method for collision detection [Teschner et al. 2004]). If a bottom-up approach is used, one has to take care that only adjacent triangles should be merged and the resulting region should be as "well-shaped" as possible. Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann 2000b] determine the "shape factor" by the ratio of $\sqrt{SurfaceArea}/ContourLength$, which is small for "well-shaped" areas. Mezger et al. [Mezger et al. 2003] used a top-down approach where the $k$-DOP is split according to it's longest side. Provot [Provot 1997] also uses a top-down approach, whereas Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann 1994] use a region-merge algorithm to build the hierarchy bottom-up.

To detect collisions the hierarchy is traversed top-down and pairs of nodes are recursively tested for overlap. If the nodes are leaves then the enclosed primitives are tested for intersection. The collision itself can be of two type [Provot 1997]: either a node of one mesh went through a triangle of another mesh or the edge of a triangle of one mesh went through another edge of the other mesh. To implement the collision detection properly, the numerical integration method of the simulation has to be taken into account. For example, for an Euler method the velocity of a vertex is constant during an interval $[t_0, t_0 + \Delta t]$. This holds not true e.g. for Runge-Kutta methods, where the velocity is evaluated at several times within the interval $[t_0, t_0 + \Delta t]$.

### 4.2 Self-intersections

Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann 1994] suggested an exact method to reject possible self-intersections. They observe that self-intersections of a continuous surface $S$ delimited by one contour $C$ are not possible if there exists a vector $\mathbf{V}$ for which $\mathbf{NV} > 0$ at (almost) every point of $S$ and the projection of $C$ on a plane orthogonal to $\mathbf{V}$ along the direction of $\mathbf{V}$ has no self-intersections. This means if a vector can be found for an area, for which the dot product is positive for every triangle of

[2] A velocity cone is similar to a normal cone, which is discussed in Section 4.2. Instead of normals, the velocities of the vertices are considered.

the area and the projected contour does not self-intersect, this area can be discarded from the collision detection algorithm.

Provot [Provot 1997] introduced the idea of a "normal cone", which was inspired by the work of Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann 1994]. He evaluates the curvature of a zone (part of a surface) by the set of normals of the triangles belonging to the zone. A cone, containing all the normals, is constructed and if the opening angle $\alpha$ of the cone satisfies $\alpha < \pi$ the zone cannot self-intersect. Note that $\alpha \geq \pi$ does not mean that there exists a self-intersection, rather than the possibility of it. As noted by several other authors (e.g. [Mezger et al. 2003]) this method can fail if a zone has a concave shape. Mezger et al. [Mezger et al. 2003] extended the concept to detect proximities. In this case the above condition must be replaced by

$$\alpha \geq \varepsilon_{closeAngle} \leq \pi \tag{14}$$

whereas the exact choice of $\varepsilon_{closeAngle}$ is not crucial, as noted by the authors. Decreasing the angle allows for spiky bends of the cloth.

### 4.3 Collision response

Once collisions have been detected they must be resolved properly. Choi and Ko [Choi and Ko 2003] referred to it as one of the *thorniest* problems in cloth simulation, because resolving one collision can cause new secondary collisions. Differently stated, the system is not guaranteed to reach a collision free state after a single resolution step. This can be circumvented by testing the result of a resolution step iteratively, which may take an inefficient amount of time and is not guaranteed to converge. Therefore, Provot [Provot 1997] proposed the grouping of particles involved in multiple collisions. These groups are handled as rigid bodies, which is justified by observing that when cloth bunches together, friction will prevent most relative motions [Bridson et al. 2002]. It must be stated that sophisticated collision response handling, as for example presented in [Bridson et al. 2002] or [Baraff et al. 2003], is not suitable for real-time applications.

Collisions are resolved by enforcing constraints on the vertices/particles of the cloth (cloth/rigid body) or by adding penalty forces (cloth/cloth). As already addressed in Section 3.2 the inverse mass of a particle can be used to constrain the velocity of this particle. The velocity can also be constrained in only one or two dimensions, by not thinking of the mass of a particle as a scalar but rather as a $3 \times 3$ inverse mass matrix, as done by Baraff et al. [Baraff and Witkin 1998]. This inverse mass matrix is given by $(1/m_i)\mathbf{I}$ where $\mathbf{I}$ is the $3 \times 3$ identity matrix. These constrained particles can be incorporated into Equation 5. For that purpose the inverse mass matrix $\mathbf{M}$ must be modified to allow inverse $3 \times 3$ matrices as masses. Equation 5 can then be rewritten as

$$\Delta\mathbf{v} = h\mathbf{W}(\mathbf{f}_0 + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} h(\mathbf{v}_0 + \Delta\mathbf{v}) + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \Delta\mathbf{v}) + \mathbf{z} \tag{15}$$

where $\mathbf{z}$ is the change in velocity which should be enforced on the particle's constrained directions. Such constrains are also used by [Provot 1997; Bridson et al. 2002; Baraff et al. 2003; Choi and Ko 2005].

Eberhardt et al. [Eberhardt et al. 1996] presented a method which introduces no additional penalty forces. Their method proved to be very efficient and well suited for real-time applications. Let $\mathbf{p}_0$ be the initial position of a particle and $\mathbf{p}_1$ the new position after an iteration of the simulation. When a collision of the particle occurs at a point $\mathbf{p}$ with normal $\mathbf{n}$ the velocities of the particles are modified. To prevent further penetration, only velocities with no direction component opposite to $\mathbf{n}$ at $\mathbf{p}$ are allowed. Additionally the position of the particle is set to $\mathbf{p}' = \mathbf{p} + d\mathbf{n}$, where $d$ is the distance between $\mathbf{p}$ and $\mathbf{p_1}$. The new velocity is

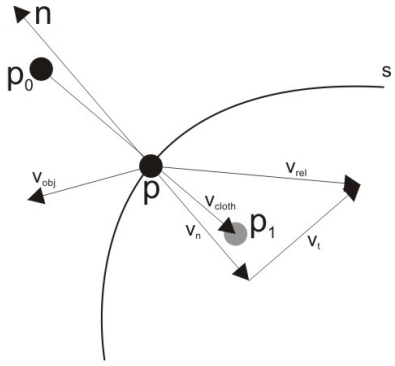


Figure 7: A particle with velocity $\mathbf{v}_{cloth}$ collides with a surface $s$ with velocity $\mathbf{v}_{obj}$ at point $\mathbf{p}$.

$$\mathbf{v}' = c_{fric}\mathbf{v}_t + c_{refl}\mathbf{n}\|\mathbf{v}_n\| \tag{16}$$

where $\mathbf{v}_n$ is the component of the initial velocity parallel to the intersection normal $\mathbf{n}$ and $\mathbf{v}_t$ is the component perpendicular to $\mathbf{n}$. $c_{fric}$ is the friction coefficient and $c_{refl}$ the reflection coefficient of the material. These coefficients can be derived for cloth from the Kawabata plots which are shortly discussed in Section 2. Eberhardt et al. [Eberhardt et al. 1996] handled collisions of the cloth with non-moving objects. Vassilev et al. [Vassilev et al. 2001] modified the method for the dynamic case, when both objects are moving. In this case the relative velocity between the cloth and the object has to be computed as $\mathbf{v}_{rel} = \mathbf{v}_{cloth} - \mathbf{v}_{obj}$. $\mathbf{v}_n$ and $\mathbf{v}_t$ are then derived from $\mathbf{v}_{rel}$. Figure 7 shows a geometrical representation of the forces involved in the above calculation.

## 5 GPU Implementation

Due to the increasing processing power of modern GPUs they can be efficiently used to take over tasks from the CPU. Since cloth can be modeled as a 2D network of particles, the simulation of cloth is well suited to a GPU implementation. They also opened the door for fast image-based collision detection.

### 5.1 Simulation

The first cloth simulation algorithm running on a GPU was presented by Green [Green 2004]. The drawbacks of his implementation have already been mentioned in the introduction. In [Zeller 2006] an improved version is presented, which will be discussed in what follows. The method is based on the work of Jakobsen [Jakobsen 2003] which was described in Section 3.3. Cloth is modeled as a set of particles connected with springs. Only structural and shear springs are considered.

The current and previous positions are stored in floating point textures, as well as the new positions. This is necessary because reading from and writing to the same texture is not permitted on graphics hardware. In case of a rectangular piece of cloth a direct mapping to a texture is possible. For triangular meshes, Zeller [Zeller

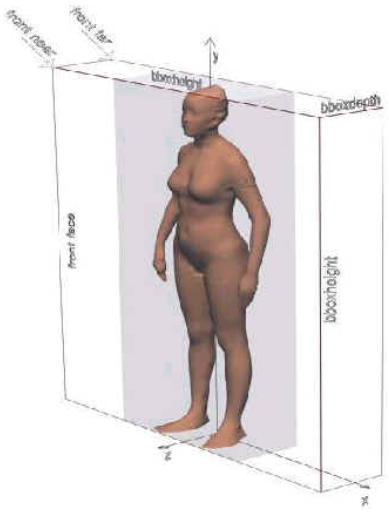


Figure 8: A human body enclosed by it's bounding box. A orthogonal camera is placed at the front and back face to obtain the depth maps. Image from [Vassilev et al. 2001].

2006] proposed remeshing the triangular meshes into a 2D array of quantized points as described in [Gu et al. 2002]. The constraints are enforced at the end of a simulation step by relaxation. Zeller [Zeller 2006] points out that it is important that two constraints with one particle in common get enforced sequentially for the relaxation algorithm to converge. Therefore, the result of the first constraint is stored into a texture, which can then be accessed by the fragment shader responsible for the second constraint. In case of the above configuration eight rendering passes are necessary (four for the structural springs and four for the shear springs). The relaxation is stopped after a certain number of iterations. However, this can lead to useless iterations if the solution has already converged to a state in which the deformation of the cloth has no significance any more.

Therefore, Rodriguez-Navarro et al. [Rodriguez-Navarro et al. 2005] uses a *quasi feedback* method to stop the iterative process saving a lot of unnecessary iterations. They invoke an occlusion query and render the cloth discarding the points whose displacement are less than a certain threshold. The result of the occlusion query is the number of samples below that threshold. Depending on that value the relaxation is either stopped or continued. Rodriguez-Navarro et al. [Rodriguez-Navarro et al. 2005] also proposed the use of a connectivity texture for arbitrary triangular meshes. The connectivity texture stores the indices of the neighbors and the rest length of the associated spring. For a mesh in which every vertex has a maximum of $N_{max}$ neighbors, the connectivity texture is of size $(wN_{max}, h)$. Since the number of neighbors is usually not constant the texture is initialized with $-1$ to mark when no more neighbors exist. The indexes of a vertex $p_{ij}$ are then at positions $(i*N_{max}, j)$, $(i*N_{max}+1, j)$,... ,$(i*N_{max}+(N_{max}-1), j)$.

## 5.2 Collision Detection and Response

Using an image-based method it is possible to detect collisions by comparing the depth map of one object with the depth map of another object. Vassilev et al. [Vassilev et al. 2001] applied an image-based collision method – which formally have mostly been used to detect rigid-body interferences – to cloth-body collision detection and response. For rendering the depth maps they placed two orthogonal cameras at the center of the front and back face of the body's bounding box pointing to it's center (see Figure 8). To increase the accuracy of the depth map, they set the far clipping plane to the face farther away and the near clipping plane to the face near the camera. To test for collisions of a cloth point $(x_w, y_w, z_w)$ with the body, the $(x_w, y_w)$ values must be converted from world-space to the map-coordinate system, yielding coordinates $(x_{d_{front}}, y_d)$ and $(x_{d_{back}}, y_d)$. Afterwards the $z_w$-value is used to decide which of the two depth maps to use. Equation 17 shows the comparison for the front and back depth map respectively.

$$\begin{aligned} front &: z_w > depthmap(x_{d_{front}}, y_d) \\ back &: z_w < depthmap(x_{d_{back}}, y_d) \end{aligned} \tag{17}$$

Since only two sides of the human body are rendered into depth maps some collisions can arise which can not be extracted from these depth maps. Therefore, Rodgriguez-Navarro et al. [Rodriguez-Navarro et al. 2005] consider several detailed views of the anatomy. They break down the humanoid into *anatomical sets* (head, torso, 3 segments for each arm and leg). For every set two orthogonal cameras are placed (as described above) which gives a total of 28 depth maps.

Both also used the GPU to handle collision response. During the rendering of the depth maps, normal maps and velocity maps can be computed by substituting the $(r,g,b)$ components with the components of the corresponding vector. Since color values are in the range $[0,1]$ the vectors must be converted appropriately. To improve performance the depth map and the normal map are stored in one RGBA texture [Rodriguez-Navarro et al. 2005].

Wong [Wong 2004] used image-based collision detection for tracking self-intersections of the cloth for which they resorted to Provot's normal cone. They speak of a $(\pi,\beta)$-surface instead of a normal cone, but conceptually it is the same. $\pi$ is the surface direction and $\beta < \pi/2$ is the surface angle. As Provot, Wong [Wong 2004] is interested in $(\pi,\beta)$ surfaces which are as large as possible. This is accomplished by applying a hierarchical approach similar to the one used by Provot [Provot 1997].

To partition a surface into $(\pi,\beta)$-surfaces a single triangle[3] is used as starting point and all of its adjacent triangles are stored in a queue[4]. At each step a triangle is chosen from the queue and checked if it can be added to the current surface. If it can, $\pi$ and $\beta$ are updated and the edge-shared triangles are added to the queue if they do not belong to another $(\pi,\beta)$-surface. Otherwise a new surface is constructed. The process is repeated until every triangle belongs to a $(\pi,\beta)$-surface. Two surfaces $(\pi_0,\beta_0)$ and $(\pi_1,\beta_1)$ can be merged if $(\beta_0+\beta_1+\gamma) < 2\beta_{max}$, where $\gamma$ is the angle between $\pi_0$ and $\pi_1$. The new surface direction is simple the mean vector of $\pi_0$ and $\pi_1$ and the new surface angle is computed as[5] $\beta = (\beta_0+\beta_1+\gamma)/2$ (opposed to Provot [Provot 1997] who used $\beta = \gamma/2 + max(\beta_0,\beta_1)$).

Following the explanation of Wong [Wong 2004] each surface is orthogonally projected onto a viewport along $\pi$, whereas each triangle is assigned a unique color. The contents of the frame buffer are read back and are stored into a 2-dimensional array. If the colors of two adjacent pixels correspond to triangles which do not share a vertex, a collision may occur. In this case a triangle intersection test

[3] A single triangle is by definition a $(\pi,\beta)$-surface, where $\pi$ is the normal of the triangle and $\beta = 0$.

[4] That way the nearest triangles are merged first.

[5] This equation has been advocated by Sederberg and Meyers [Sederberg and Meyers 1988].

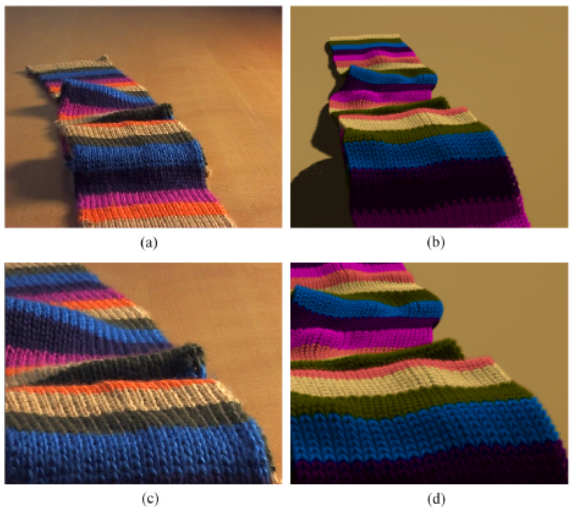


Figure 9: Comparision of photographs to lumislice-based rendering. Images (a) and (c) show real images and (b) and (d) shows the corresponding renderings. Image reprinted from [Xu et al. 2001].

is performed. To detect collisions among different $(\pi,\beta)$-surfaces a bounding volume tree is constructed for each surface. Once the trees are constructed, they can be traversed for each pair of surfaces to detect possible triangle collisions.

## 6 Rendering

To render textiles in a sophisticated way the internal structure of the cloth must be taken into account. This poses a major challenge because the light reflection can not be handled with simple texture mapping and modeling the individual stitches and knits geometrically is out of question due to the sheer complexity. Especially the rendering of knitwear is a considerable challenge, because its thickness requires volume rendering [Xu et al. 2001]. For woven fabrics specialized BRDF models exist, like the one presented in [Yasuda et al. 1992]. For knitware, Xu et al. [Xu et al. 2001] introduced the lumislice which produces very sophisticated results as shown in Figure 9, but is only suitable for offline rendering. In contrast Daubert et al. [Daubert et al. 2001] describe a method closely related to bidirectional texture functions (BTFs).

A $BRDF_\lambda(\theta_i,\phi_i,\theta_o,\phi_o)$, short for bi-directional reflectance distribution function, describes for a given wavelength $\lambda$ how much light is reflected in a particular outgoing direction depending on the incoming light direction. $\theta_i$ and $\phi_i$ describe the incoming light direction in spherical coordinates and $\theta_o$,$\phi_o$ the outgoing light direction, respectively. The term BTF (bidirectional texture function) was introduced by Dana et al. [Dana and Nayar 1999] and describes the appearance of a texture under arbitrary view and lighting conditions. A BTF can be written as a 6-dimensional function $f(\theta_i,\phi_i,x,y,\theta_o,\phi_o)$ where $(x,y)$ is a specific surface point of the flat sample. BTFs are high-dimensional and involve a large amount of data. For practical implementations on graphics hardware a compact representation is needed. For a description of how to efficiently store BTFs see [Sattler et al. 2003]. Daubert et al. [Daubert et al. 2001] use a spatially varying BRDF representation for the material. It is assumed that the material consists of one or a small number of stitch types, which are repeated over the garment. Each stitch is represented as a geometric model and sampled regularly within a plane into a 2D array. For each entry the parameters of a Lafortune reflection model, a look up table, the normal and the tangent are stored. The BRDF $f(\mathbf{l},\mathbf{v})$ for an entry is given by

$$f(\mathbf{l},\mathbf{r}) = T(\mathbf{v}) f_l(\mathbf{l},\mathbf{v}) \tag{18}$$

where $f_l(\mathbf{l},\mathbf{v})$ denotes the Lafortune model [Lafortune et al. 1997] and is defined as:

$$f_l(\mathbf{l},\mathbf{v}) = \rho + \sum_i [C_{x_i} v_x l_x + C_{y_i} v_y l_y + C_{z_i} v_z l_z]^{N_i} \tag{19}$$

where $\rho$ is the diffuse part. The parameter $C_{x_i}, C_{y_i}, C_{z_i}$ and $N_i$ define the size and shape of the $i$th lobe. The lookup table stores color and alpha values for each of the viewing directions. The rendering process consists of four steps [Daubert et al. 2001]: 1. interpolate the per-pixel normals. 2. Compute indices into the pattern array, yielding a BRDF $f_r$. 3. Evaluate $f_r$ with light and view direction mapped into the geometry's local coordinate system. Finally, write the result into the framebuffer. Figure 10 (left) shows a rendered image of a dress obtained with this method.

Another approach for realtime shading of static meshes, based on the visibility cones of Stewart [Stewart 1999], has been published by Ganster et al. [Ganster et al. 2002]. The main idea from Stewart is to compute visibility cones for each vertex of the surface, which determine the parts of the environment seen by this vertex. By doing several intersection tests, the irradiance at a vertex can be evaluated. However, the intersection tests can get quite complicated, depending on the light source. Therefore, Ganster et al. [Ganster et al. 2002] proposed binary visibility maps. To calculate the visibility map a finite set of directions on the hemisphere belonging to a vertex are considered. For each direction the visibility of the environment is determined. If the environment is occluded by the surface, 0 is stored in the map, otherwise 1. During runtime the visibility map is used to decide whether the radiance from a certain direction contributes to the radiance of the vertex or not. The obtained values are then interpolated across the triangles to obtain the radiance for the whole mesh. Figure 10 (right) shows a dress rendered with this method. The major drawback of this approach is that the cloth itself must be static. However, as pointed out by the authors themself, an optimization would be to only update the binary visibility map of vertices, whose position have changed.

## 7 Conclusion

The process of representing cloth can roughly be divided into three steps. First, the cloth has to be modeled and simulated according to some physically-based simulation (or geometric method). Second, collisions must be detected and resolved in a stable and fast way, which is the major bottleneck for cloth animation. Third, the textile should be rendered in a sophisticated way. This means that the underlying microstructure of the fabric must be taken into account. Since the first approaches in the second half of the 1980s many different techniques have been developed. These techniques either aim at high-quality offline rendering or at fast and physically plausible real-time rendering. Programmable GPUs allow the execution of some or all parts of the simulation. If the result of the simulation must not be available on the CPU and is only used for rendering, GPU implementations are from even greater importance.

Figure 10: left: A dress rendered with a spatially varyiing BRDF consisting of one lobe (from [Daubert et al. 2001]). right: A folded dress consisting of 3120 vertices rendered with the help of binary visibility maps (reprinted from [Ganster et al. 2002]).

## A Integration Methods

The numerical stability of cloth simulation methods relies on the used integration scheme. Explicit integration schemes are the simplest methods available, e.g. Euler method or the family of Runge-Kutta methods, for solving first-order differential equations. The explicit Euler method

$$x_{k+1} = x_k + \Delta t f(t_k, x_k) \tag{20}$$

where $f(t_k, x_k) = \dot{x}_k$ calculates the new state from the current state and its first derivative. Since this method takes no notice of wildly changing derivatives,numerical errors are introduced into the solution. A well chosen time step $\Delta t$ is therefore crucial for a reasonable numerical stability and depends on the stiffness of the system. Making the time step small yields longer simulation time and is therefore not suitable. Therefore Provot as well as Vassilev et al. [Provot 1995; Vassilev et al. 2001] used the fact that the Euler method is very stable if the time step is less than the natural period of the system

$$\Delta t \leq \pi \sqrt{\frac{m}{K}} \tag{21}$$

where $K$ is the highest stiffness of the system. Vassilev et al. [Vassilev et al. 2001] showed experimentally that for the system to be numerically stable even $\Delta t \leq 0.4\pi\sqrt{m/K}$ must hold true. However, the resulting time steps can be quite small. Adaptive time stepping methods – taking into account the current simulation error – have also been proposed (see e.g. [Volino and Magnenat-Thalmann 2000b]). Another possibility would be to use a more advanced explicit integration scheme (e.g. second-order midpoint, fourth-order Runge-Kutta) at the cost of computational speed.

For this reasons Baraff and Witkin [Baraff and Witkin 1998] departed from explicit integration methods and used an implicit Euler method

$$x_k = x_{k+1} + \Delta t f(t_{k+1}, x_{k+1}) \tag{22}$$

which introduces an additional layer of consistency, since the derivative at least points back to where you come from. Implicit integration has also been used by Terzopoulos et al. [Terzopoulos et al. 1987; Terzopoulos and Witkin 1988] but seemed to have vanished until the work of Baraff. Bridson et al. [Bridson et al. 2003] use a mixed explicit/implicit time integration to combine the simplicity of explicit integration methods with the efficiency of implicit methods (the implementation difficulty lies in finding the Hessian matrix for the system). A comparison of integration methods in regard to cloth simulation can be found in [Volino and Magnenat-Thalmann 2001].

## References


BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. *Computer Graphics 32*, Annual Conference Series, 43–54.

BARAFF, D., WITKIN, A., AND KASS, M., 2003. Untangling cloth.

BREEN, D. E., HOUSE, D. H., AND GETTO, P. H. 1992. A physically-based particle model of woven cloth. *The Visual Computer 8*, 264–277.

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

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. 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*, ACM Press, New York, NY, USA, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 28–36.

CARAMANA, E. J., BURTON, D. E., SHASHKOV, M. J., AND WHALEN, P. P. 1998. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *J. Comput. Phys. 146*, 1, 227–262.

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

CHOI, K., AND KO, H. 2003. Research problems in clothing simulation. *Computer-Aided Design*.

CHOI, K.-J., AND KO, H.-S. 2005. Stable but responsive cloth. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, ACM Press, New York, NY, USA, 1.

DANA, K. J., AND NAYAR, S. K. 1999. 3d textured surface modeling. *IEEE Workshop on the Integration of Appearance and Geometric Methods in Object Recognition*, 46–56.

DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. 1999. Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1, 1–34.

DAUBERT, K., LENSCH, H. P. A., HEIDRICH, W., AND SEIDEL, H.-P. 2001. Efficient cloth modeling and rendering. In *Proceedings of the Eurographics Workshop on Rendering*. To appear.

EBERHARDT, B., WEBER, A., AND STRASSER, W. 1996. A fast, flexible, particle-system model for cloth draping. *IEEE Comput. Graph. Appl. 16*, 5, 52–59.

FUNFZIG, C., AND FELLNER, D. W. 2003. Easy realignment of k-dop bounding volumes. *Proceedings of Graphics Interface*, 257–264.

GANSTER, B., KLEIN, R., SATTLER, M., AND SARLETTE, R. 2002. Realtime shading of folded surfaces. *CGI Proceedings*.

GREEN, S., 2004. Cloth simulation. available online: `http://developer.nvidia.com/object/demo_cloth_simulation.html`.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 355–361.

HOUSE, D. H., AND BREEN, D. E. 2000. *Cloth Modeling and Animation*. AK Peters.

JAKOBSEN, T. 2003. Advanced character physics. available online: `http://www.gamasutra.com/resource_guide/20030121/jacobson_01.shtml`.

KAWABATA, S., NIWA, M., AND KWAI, H. 1973. The finite deformation theory of plain weave part i to iii. *Journal of the Textile Institue*.

KLOSOWSKI, J. T., HELD, M., MITCHELL, J. S. B., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics 4*, 1, 21–36.

LAFORTUNE, E. P. F., FOO, S.-C., TORRANCE, K. E., AND GREENBERG, D. P. 1997. Non-linear approximation of reflectance functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 117–126.

MEZGER, J., KIMMERLE, S., AND ETZMUSS, O. 2003. Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG 11*, 2, 322–329.

PEIRCE, F. T. 1937. The geometry of cloth structure. *Journal of the Textile Institute 28*.

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

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design harments. *Proc. Computer Animation and Simulation*, 177–189.

RODRIGUEZ-NAVARRO, J., SAINZ, M., AND SUSN, A. 2005. Gpu based cloth simulation with moving humanoids. *Actas XV Congreso Espaol de Informtica Grfica (CEIG'2005)*, 147–155.

SATTLER, M., SARLETTE, R., AND KLEIN, R. 2003. Efficient and realistic visualization of cloth. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 167–177.

SEDERBERG, T. W., AND MEYERS, R. J. 1988. Loop detection in surface patch intersections. *Comput. Aided Geom. Des. 5*, 2, 161–171.

SHEWCHUK, J. R. 1994. An introduction to the conjugate gradient method without the agonizing pain. Tech. rep., Pittsburgh, PA, USA.

STEWART, A. J. 1999. Computing visibility from folded surfaces. *Computers and Graphics 23*, 5 (October), 693–702.

TERZOPOULOS, D., AND WITKIN, A. 1988. Deformable models. *IEEE Computer Graphics and Applications 8*, 6 (November), 41–51.

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

TESCHNER, M., KIMMERLE, S., ZACHMANN, G., HEIDELBERGER, B., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNETAT-THALMANN, N., AND STRASSER, W. 2004. Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)*, Eurographics Association, Eurographics Association, 119–139.

VASSILEV, T., SPANLANG, B., AND CHRYSANTHOU, Y. 2001. Fast cloth animation on walking avatars. *Computer Graphics Forum 20*.

VOLINO, P., AND MAGNENAT-THALMANN, N. 1994. Efficient self-collision detection on smootly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum 13*, 3, 155–166.

VOLINO, P., AND MAGNENAT-THALMANN, N. 1997. Interactive cloth simulation: Problems and solutions. *Proceedings of the JWS*.

VOLINO, P., AND MAGNENAT-THALMANN, N. 1997. Developing simulation techniques for an interactive clothing system. In *VSMM '97: Proceedings of the 1997 International Conference on Systems and MultiMedia*, IEEE Computer Society, Washington, DC, USA, 109.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2000. Implementing fast cloth simulation with collision response. In *CGI '00: Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, Washington, DC, USA, 257.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2000. *Virtual Clothing*. Springer.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2001. Comparing efficiency of integration methods for cloth simulation. In *CGI '01: Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, Washington, DC, USA, 265.

VOLINO, P., COURCHESNE, M., AND THALMANN, N. M. 1995. Versatile and efficient techniques for simulating cloth and other deformable objects. In *SIGGRAPH '95: Proceedings of the 22nd*

*annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 137–144.

WANG, Y., 2005. Gpu based cloth simulation on moving avatars.

WEIL, J. 1986. The synthesis of cloth objects. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 49–54.

WONG, W. S.-K. 2004. Image-based collision detection for deformable cloth models. *IEEE Transactions on Visualization and Computer Graphics 10*, 6, 649–663. Member-George Baciu.

XU, Y.-Q., CHEN, Y., LIN, S., ZHONG, H., WU, E., GUO, B., AND SHUM, H.-Y. 2001. Photorealistic rendering of knitwear using the lumislice. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 391–398.

YASUDA, T., YOKOI, S., ICHIRO TORIWAKI, J., AND INAGAKI, K. 1992. A shading model for cloth objects. *IEEE Comput. Graph. Appl. 12*, 6, 15–24.

ZELLER, C. 2005. Cloth simulation on the gpu. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, ACM Press, New York, NY, USA, 39. available online: `http://download.nvidia.com/developer/presentations/2005/SIGGRAPH/ClothSimulationOnTheGPU.pdf`.

ZELLER, C. 2006. *Shader X4: Advanced Rendering Techniques*. Charles River Media, ch. Practical Cloth Simulation on Modern GPUs, 17–27.

ZHANG, D., AND YUEN, M. M. F. 2000. Collision detection for clothed human animation. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 328.