Fast and stable cloth simulation based on multi-resolution shape matching

Jan Bender[a], Daniel Weber[b], Raphael Diziol[c]

[a] *Graduate School CE, TU Darmstadt*
[b] *Fraunhofer IGD, Darmstadt*
[c] *Karlsruhe Institute of Technology*

Abstract

We present an efficient and unconditionally stable method which allows the deformation of very complex stiff cloth models in real-time. This method is based on a shape matching approach which uses edges and triangles as 1D and 2D regions to simulate stretching and shearing resistance. Previous shape matching approaches require large overlapping regions to simulate stiff materials. This unfortunately also affects the bending behavior of the model. Instead of using large regions, we introduce a novel multi-resolution shape matching approach to increase only the stretching and shearing stiffness. Shape matching is performed for each level of the multi-resolution model and the results are propagated from one level to the next one. To preserve the fine wrinkles of the cloth on coarse levels of the hierarchy we present a modified version of the original shape matching method. The introduced method for cloth simulation can perform simulations in linear time and has no numerical damping. Furthermore, we show that multi-resolution shape matching can be performed efficiently on the GPU.

Keywords: cloth simulation, shape matching, multi-resolution model, GPU-based simulation

1. Introduction

Interactive simulation of cloth has been an important research topic in computer graphics for many years. An interactive simulation demands efficient and unconditionally stable methods.

In the beginning simple spring-mass systems in combination with explicit integration schemes were used in order to obtain a high performance. The problem is that either such models generally stretch significantly under their own weight or the high performance is lost since a small time step size is required for a stable simulation of stiff materials. However, a stiff model is required to obtain realistic results. In the following years implicit integration methods were investigated in order to perform an efficient and stable simulation of stiff materials. These integration methods are unconditionally stable and therefore allow to perform larger time steps. The drawback of using implicit integration is that a large non-linear or a linear system has to be solved in each step. Furthermore, popular methods like the implicit Euler introduce numerical damping which is not desired.

In this paper we present a geometrically motivated method based on shape matching [1] to simulate the physically plausible deformation of a cloth model. Shape matching is simple to implement, unconditionally stable and very efficient. Moreover, it does not introduce numerical damping to the system. Shape matching approaches usually subdivide the particles of a model into several overlapping regions. For each region the initial configuration of the model is matched to the current deformed state in order to define goal positions. Then the particles are pulled towards this goal configuration to simulate elasticity. The global stiffness of the model depends on the region sizes. The simulation of cloth requires a high stretching and shearing stiffness while the bending resistance is low. The problem is that region-based approaches cannot differentiate between stretching, shearing and bending. Furthermore, the computation time for the simulation of arbitrary meshes depends not only on the size of the mesh but also on the region size.

Figure 1: Dancing Woody. The finest mesh of the multi-resolution cloth model consists of 32467 triangles whereas the total number of triangles in all five hierarchy levels is 62888. The simulation of the deformation requires 3.89 ms per step on average.

Therefore, we introduce a novel shape matching

method which uses a multi-resolution model instead of large overlapping regions. Multi-resolution shape matching allows us to simulate realistic fabrics with a high stretching and shearing stiffness. Moreover, this novel approach is very fast since the computation time depends linearly on the size of the cloth model. The use of a multi-resolution model has also the advantage that information is propagated very fast through a mesh. This permits more realistic results especially for high resolution models.

Our contributions:

- An efficient cloth simulation method based on shape matching and an analysis of its performance compared to standard methods.
- A multi-resolution approach to increase the stretching and shearing stiffness of the cloth model by performing shape matching on different resolution levels of the model.
- A method to conserve the wrinkles of the finest mesh on the coarser levels of the multi-resolution hierarchy.
- An efficient GPU implementation of the simulation method.

2. Related Work

The physically-based simulation of cloth models has a long history in computer graphics. Surveys are given by Thalmann and Volino [2] as well as Nealen et al. [3].

In the first years of research many works treated cloth as an elastic material and used explicit integration methods to solve the equation of motion, e.g. [4, 5]. The problem is that many real fabrics are not very stretchable and explicit integration methods have to use a very small step size to simulate stiff materials without getting numerically instable [6]. Therefore, Provot [5] proposed to manipulate the vertex positions of the cloth model after each simulation step in order to get a stiffer material. Later, implicit methods which are unconditionally stable became popular [7, 8, 9] since these methods allow to perform larger time steps for stiff models. While in the beginning discrete models were typically used to simulate cloth, later continuous models were also investigated since such models provide a more accurate representation. Etzmuss et al. [10] presented an efficient simulation approach for arbitrary triangle meshes based on a linear finite element method (FEM) with a corotational formulation. Volino et al. [11] introduced a cloth simulation system based on continuum mechanics. This system is able to simulate nonlinear anisotropic materials.

Instead of treating cloth as an elastic material different works introduce methods to simulate totally inextensible cloth by using hard constraints [12, 13, 14]. Goldenthal et al. [12] introduced a method based on constrained Lagrangian mechanics in combination with a fast projection to simulate inextensible cloth efficiently. English and Bridson [14] simulated inextensible cloth by using a triangle mesh with hard distance constraints. They solved the locking problem which occurs with such a simulation model by using a nonconforming mesh for the simulation.

Strain limiting is an important research topic in the area of cloth simulation. The goal of strain limiting is to guarantee some limits for the strain of the cloth model. Cloth is treated as an elastic material until a certain maximum strain and then hard constraints are used to limit the strain. Therefore, it is a kind of combination between methods for elastic and inextensible materials. Continuum-based strain limiting was introduced by Thomaszewski et al. [15]. Wang et al. [16] proposed a strain limiting approach for finite element simulations. They first compute the deformation gradient for each element of the model. Then a singular value decomposition is performed to get the principal strains. In order to limit the strain of the model, the principal strains are clamped and a new deformation gradient is constructed. In this way strain limiting is coordinate-independent. Finally, the positions of the triangle vertices are corrected using the new deformation gradient. Global strain limiting is performed using Gauss-Seidel iteration. Bridson et al. [17] limited the strain and the strain rate of their cloth model by impulses instead of manipulating the positions directly in order to prevent self-penetrations.

In the last years, position-based simulation methods for deformable objects became popular. A survey on this topic is given by Bender et al. [18]. Position-based approaches are unconditionally stable, controllable and faster than traditional simulation methods. However, position-based methods are generally not as accurate as traditional ones but provide visual plausibility. Müller et al. [1] proposed a meshless method for physically plausible deformations based on shape matching. Shape matching has the advantages that it is very fast and unconditionally stable. In order to simulate large deformations Müller et al. proposed to perform shape matching for overlapping regions and to combine the results. In each time step shape matching is performed just once per region for the sake of performance. A maximum strain cannot be guaranteed in this way and only elastic materials without strain limit can be simulated. However, increasing the region size allows to simulate very stiff materials. Later, region-based shape matching was optimized for lattices [19, 20] and incompressible surfaces [21]. Recently, an extension of the original shape matching method was provided by Müller and Chentanez [22]. They used so-called oriented particles to add rotational information to each particle. In this way shape matching gets robust for sparse structures. This kind of methods perform shape matching for overlapping regions to allow large deformations where the stiffness of the model depends on the size of the regions. The position-based dynamics method of Müller et al. [23] solves constraints on the position level in order to simulate deformations. In [24] a multi-resolution approach was introduced to speed up the convergence of their constraint

solver. Müller et al. use only simple one-dimensional distance constraints to simulate the in-plane stiffness of their cloth models. In contrast to them we introduce 2D shape matching which simulates stretching resistance in warp and weft direction as well as shearing resistance. Moreover, in our work we do not have to solve a linear system. We use the multi-resolution model to enlarge the influence of the shape matching regions which results in a higher stiffness.

Stumpp et al. [25] presented an efficient approach for cloth simulation based on region-based shape matching. They define a region for each triangle by taking the adjacent triangles into account. In this way they get overlapping regions with three vertices. The vertices of the adjacent triangles are generally not coplanar. Therefore, this approach induces out-of-plane forces which influence the bending stiffness and depend on the size of the model. In general, this behavior is undesired in cloth simulation. Another problem of this approach is that a high stretching and shearing stiffness cannot be obtained for large models due to small shape matching regions. The authors add additional fiber clusters to increase the stiffness, but this cannot guarantee a high stiffness for large models.

In contrast to the approach of Stumpp et al., we use a multi-resolution model in this paper to solve the problem of low stretching and shearing stiffness. Furthermore, we propose novel shape matching methods and show how fine wrinkles are conserved with our approach.

3. Overview

In this section we want to give a short overview of our cloth simulation method before we discuss each step in detail.

Our cloth model is a triangular mesh of particles where each particle has a mass m, a position $\mathbf{x}$ and a velocity $\mathbf{v}$. The masses of the particles are determined by assuming a constant density ρ of the material. Using this assumption, the mass of a particle i is approximated by

$$m_i = \rho \cdot a, \quad a = \frac{1}{3} \sum_{j \in \mathrm{tri}(i)} a_j, \tag{1}$$

where $\mathrm{tri}(i)$ is the set of adjacent triangles of the vertex i and a_j is the area of the triangle j. To normalize the weights of the particles during the shape matching process, we also define the mass per region $\tilde{m}_i = m_i / |\mathfrak{R}_i|$ as proposed in [19] where $|\mathfrak{R}_i|$ is the number of shape matching regions which contain particle i.

A simulation step for this cloth model is performed as follows:

1. Determine the sum $\mathbf{F}_i^n$ of all external and internal forces (see section 4.4) acting on particle i.
2. Advance particle positions and velocities to get $\mathbf{x}_i^{n+1}$ and $\mathbf{v}_i^{n+1}$.
3. Simulate cloth model using multi-resolution shape matching (see sections 4.1 – 4.3).
4. Perform proximity detection for $\mathbf{x}_i^n$ and apply repulsion impulses with friction [17].
5. Perform continuous collision detection and resolve all collisions [17].
6. Damp velocities of the particles (see section 4.4).

The time integration of the particle positions and velocities in step 2 can be performed with any integration method of your choice. In this work we use the symplectic Euler method:

$$\begin{aligned} \mathbf{v}_i^{n+1} &= \mathbf{v}_i^n + \Delta t \frac{\mathbf{F}_i^n}{m_i} \\ \mathbf{x}_i^{n+1} &= \mathbf{x}_i^n + \Delta t \mathbf{v}_i^{n+1}. \end{aligned} \tag{2}$$

4. Cloth simulation

In this section we will introduce a region-based shape matching method to simulate complex cloth models efficiently. The advantage of the shape matching approach is that it is unconditionally stable and very fast. We use 2D and 1D regions for the simulation. With the 2D regions we can model shearing and stretching stiffness. The 1D regions are used to increase the stretching stiffness of the cloth model even more at low computational effort.

In order to prevent out-of-plane forces, shape matching must be performed in the two-dimensional space of the surface of the cloth model. The cloth model is represented as triangle mesh for the simulation. For 2D shape matching we define a region for the three vertices of each triangle while the 1D method defines a region for each edge. With these regions we avoid out-of-plane forces. However, since the stiffness strongly depends on the size of the regions when using shape matching, the stretching and shearing stiffness of the simulated cloth model will be low for complex models. Instead of using larger regions, we introduce a multi-resolution approach in section 4.3 to solve this problem.

4.1. 2D Shape Matching

We need the initial configuration of a region to perform shape matching. This configuration is defined by the initial positions $\mathbf{x}_i^0$ of the region particles. The 3D shape matching approach minimizes the term

$$\sum_i \tilde{m}_i \left(\mathbf{R} \left(\mathbf{x}_i^0 - \mathbf{t}^0 \right) + \mathbf{t} - \mathbf{x}_i \right)^2 \tag{3}$$

in order to find a rotation matrix $\mathbf{R}$ and translation vectors $\mathbf{t}$ and $\mathbf{t}^0$ which match the current and the initial configuration [1]. The resulting transformation is used to determine goal positions for all particles. The particles are pulled towards this goal configuration to simulate elasticity. The optimal translation vectors are given by the centers of mass of both configurations

$$\mathbf{t}^0 = \frac{\sum_i \tilde{m}_i \mathbf{x}_i^0}{\sum_i \tilde{m}_i}, \quad \mathbf{t} = \frac{\sum_i \tilde{m}_i \mathbf{x}_i}{\sum_i \tilde{m}_i}. \tag{4}$$

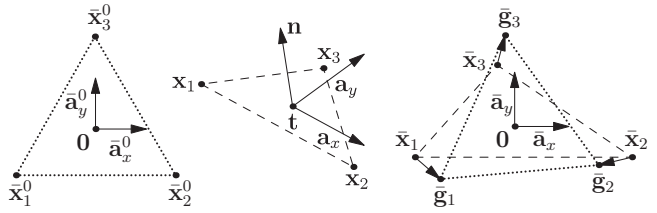


Figure 2: 2D shape matching. The undeformed initial configuration (left) is matched to the deformed configuration (middle) by first projecting both configurations in 2D and then computing the optimal rigid body transformation to get goal positions (right).

The rotational part of the affine transformation

$$\mathbf{A}_{pq} = \sum_i \tilde{m}_i \left(\mathbf{x}_i - \mathbf{t}\right)\left(\mathbf{x}_i^0 - \mathbf{t}^0\right)^T \tag{5}$$

is the optimal rotation matrix $\mathbf{R}$. 3D shape matching approaches use a 3D polar decomposition to extract the rotational part [1, 19, 21]. However, this is problematic for our triangle regions since $\det \mathbf{A}_{pq} = 0$ when all particles of a region are coplanar [21].

Triangle goal positions For 2D shape matching we can compute the translation vectors in the same way as described above. But to determine the rotation matrix, we first project the vertices into the two-dimensional space. Now we perform a 2D polar decomposition and determine the 2D goal positions of the vertices (see figure 2). In 2D a polar decomposition for our triangle regions can be performed without any problems. This is also much faster than determining the rotations in 3D. After computing the 2D goal positions we get the required position changes. Then we determine the corresponding 3D vectors and add them to the current particle positions. In the following we will mark all two-dimensional vectors and matrices with a horizontal bar to enhance the readability.

The projection of the vertices $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ of a triangle with normal $\mathbf{n} = (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)$ is performed as follows. First, we determine two orthogonal vectors in the plane of the triangle

$$\mathbf{a}_x = \frac{\mathbf{x}_2 - \mathbf{x}_1}{\|\mathbf{x}_2 - \mathbf{x}_1\|}, \quad \mathbf{a}_y = \frac{\mathbf{n} \times \mathbf{a}_x}{\|\mathbf{n} \times \mathbf{a}_x\|} \tag{6}$$

in order to get the projection matrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{a}_x^T \\ \mathbf{a}_y^T \end{pmatrix} \in \mathbb{R}^{2\times 3}. \tag{7}$$

Then, we project the vertices of the initial and the current configuration by

$$\bar{\mathbf{q}}_i = \mathbf{P}^0\left(\mathbf{x}_i^0 - \mathbf{t}^0\right), \quad \bar{\mathbf{p}}_i = \mathbf{P}\left(\mathbf{x}_i - \mathbf{t}\right), \tag{8}$$

where $\bar{\mathbf{q}}_i$ can be precomputed. The initial configuration must not contain degenerated triangles to get valid projection matrices $\mathbf{P}^0$. In the case of a degenerated triangle in the deformed configuration, we simply use the matrix $\mathbf{P}$ of the last time step to perform the projection. Note that the probability of a degenerated triangle is very low, especially when simulating stiff models. In none of our experiments, except for the test with a degenerated model, a degenerated triangle was detected.

The center of mass of a 2D triangle now lies in the origin of the 2D space. Therefore, we have to minimize

$$\sum_{i=0}^{3} \tilde{m}_i \left(\bar{\mathbf{A}}\bar{\mathbf{q}}_i - \bar{\mathbf{p}}_i\right)^2 \tag{9}$$

in order to find an optimal linear transformation $\bar{\mathbf{A}}$ for a region. The optimal transformation is determined as follows (see [1]):

$$\bar{\mathbf{A}} = \left(\sum_{i=1}^{3} \tilde{m}_i \bar{\mathbf{p}}_i \bar{\mathbf{q}}_i^T\right)\left(\sum_{i=1}^{3} \tilde{m}_i \bar{\mathbf{q}}_i \bar{\mathbf{q}}_i^T\right)^{-1} = \bar{\mathbf{A}}_{pq}\bar{\mathbf{A}}_{qq}^{-1}. \tag{10}$$

The matrix $\bar{\mathbf{A}}_{qq}^{-1}$ is symmetric and contains no rotation. We use a 2D polar decomposition $\bar{\mathbf{A}}_{pq} = \bar{\mathbf{R}}\bar{\mathbf{V}}$ to extract the optimal rotation $\bar{\mathbf{R}}$ from matrix $\bar{\mathbf{A}}_{pq}$ where $\bar{\mathbf{V}}$ is a symmetric and positive definite matrix. The rotation $\bar{\mathbf{R}}$ is determined by normalizing the column vectors of the following matrix (see [26]):

$$\bar{\mathbf{U}} = \bar{\mathbf{A}} + \operatorname{sign}(\det(\bar{\mathbf{A}}))\begin{pmatrix} \bar{A}_{22} & -\bar{A}_{21} \\ -\bar{A}_{12} & \bar{A}_{11} \end{pmatrix}. \tag{11}$$

Update particle state Now we can compute the 2D position change which is used to simulate the elastic behavior of the model

$$\bar{\mathbf{g}}_i = \bar{\mathbf{R}}\bar{\mathbf{q}}_i, \quad \Delta\bar{\mathbf{x}}_i = \alpha\frac{1}{|\Re_i|}\left(\bar{\mathbf{g}}_i - \bar{\mathbf{x}}_i\right), \tag{12}$$

where $\bar{\mathbf{g}}_i$ are the 2D goal positions of the triangle (see figure 2) and $\alpha \in [0, 1]$ is a stiffness coefficient. Since we compute the position changes for all triangle regions and a particle can be part of multiple regions, we divide by $|\Re_i|$ to get the average position change. The 3D position changes are computed by $\Delta\mathbf{x}_i = \mathbf{P}^T\Delta\bar{\mathbf{x}}_i$. Finally, the dynamic state of the particles is updated

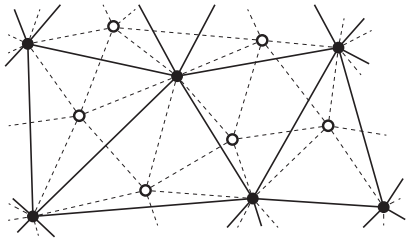
$$\mathbf{x}_i^{n+1} := \mathbf{x}_i^{n+1} + \Delta\mathbf{x}_i, \quad \mathbf{v}_i^{n+1} := \frac{\mathbf{x}_i^{n+1} - \mathbf{x}_i^n}{h}. \tag{13}$$

Since all position changes are performed in the planes of the triangles, they do not influence the bending of the model.

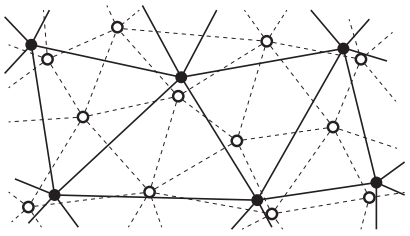
4.2. 1D Shape Matching

For 1D shape matching we define one region per edge in the cloth model consisting of the two corresponding vertices. Since we have no rotation in 1D, we only need the optimal translation $\mathbf{t}$ to perform shape matching which is defined by the center of mass (see equation (4)) of an edge. The goal positions of the vertices are

$$\mathbf{g}_1 = \mathbf{t} - q_1^0\frac{\mathbf{x}_2 - \mathbf{x}_1}{\|\mathbf{x}_2 - \mathbf{x}_1\|}, \quad \mathbf{g}_2 = \mathbf{t} + q_2^0\frac{\mathbf{x}_2 - \mathbf{x}_1}{\|\mathbf{x}_2 - \mathbf{x}_1\|}, \tag{14}$$



(a) Nested model



(b) Non-nested model

Figure 3: Two levels of a multi-resolution hierarchy. (a) Nested model: The fine mesh (dashed) contains all vertices of the coarse mesh (solid). (b) Non-nested model: The fine and the coarse mesh have no common vertex.

where $q_1^0 = \|\mathbf{x}_1^0 - \mathbf{t}^0\|$ and $q_2^0 = \|\mathbf{x}_2^0 - \mathbf{t}^0\|$ are the distances from the initial positions to the initial center of mass. These two values can be determined in a precomputation step. With the goal positions we can now compute the position change of a particle

$$\Delta\mathbf{x}_i = \beta \frac{1}{|\check{\mathfrak{R}}_i|} (\mathbf{g}_i - \mathbf{x}_i), \tag{15}$$

where $\beta \in [0,1]$ is a stiffness coefficient and $|\check{\mathfrak{R}}_i|$ is the number of 1D regions which contain the vertex i. In the end we update the dynamic state with equation (13).

Note that 1D shape matching is similar to the position-based handling of distance constraints as proposed in [23]. However, using the shape matching formulation makes the whole simulation process consistent.

4.3. Multi-Resolution Shape Matching

There are several works which use multigrid methods to solve linear systems, see e.g. [27]. But here we do not have to solve a linear system. Our goal is to increase the stretching and shearing stiffness of our cloth model. Therefore, we use a multi-resolution model which is based on a hierarchy of nested or non-nested meshes (see figure 3). Since in our model the meshes with different resolutions are coupled with each other, the influence of a shape matching region increases and the model becomes stiffer. Another advantage of using a multi-resolution model is that forces can be propagated faster through the mesh. This is an important feature especially when simulating very fine meshes with large time step sizes. The coupling is performed by two intergrid transfer operators. The prolongation operator $\mathbf{I}_l^{l+1}$ transfers values from a coarse level l to the next finer level $l+1$ and the restriction operator $\mathbf{I}_{l+1}^{l}$ transfers values from $l+1$ to the next coarser level l. For our experiments we applied a half-edge collapse operator [28] to generate the multi-resolution hierarchies. Typically we used between three and five hierarchy levels in our simulations.

In the following we first introduce multi-resolution shape matching. Then, we define the required intergrid transfer operators and show how fine wrinkles can be conserved.

Multi-resolution shape matching In a simulation step with multi-resolution shape matching we first perform the time integration for all vertices of the finest mesh. In the case of a nested model (see figure 3(a)) the positions of all particles are updated. Since vertices of a fine mesh are not part of the next coarser level for non-nested models (see figure 3(b)), we must update the vertices of all coarse levels in an additional step before we can continue. In this step we start with the finest mesh and interpolate the positions of the next coarser mesh using our restriction operator:

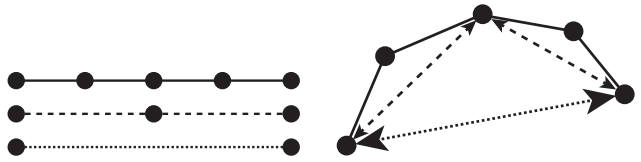
$$\mathbf{x}^{l-1} := \mathbf{I}_l^{l-1}\mathbf{x}^l.$$

After all positions are up-to-date, we perform multi-resolution shape matching for our model using the following V-cycle algorithm:

```
1 : for l = l_max to 1
2 :     Store current positions: x̂^l ← x^l
3 :     Perform shape matching
4 :     x^{l-1} := x^{l-1} + I_l^{l-1}(x^l − x̂^l)
5 : for l = 0 to l_max
6 :     Store current positions: x̂^l ← x^l
7 :     Perform shape matching
8 :     If l ≠ l_max
9 :         x^{l+1} := x^{l+1} + I_l^{l+1}(x^l − x̂^l)
```

The algorithm begins with a restriction phase (lines 1-4) which starts with the finest level l_{max}. On each level we first store the current positions and then perform a 1D and a 2D shape matching step. The resulting position differences $\mathbf{x}^l - \hat{\mathbf{x}}^l$ are projected to the next coarser level using our restriction operator $\mathbf{I}_l^{l-1}$ (line 4). In the prolongation phase (line 5-9) we go back from the coarsest mesh to the finest one performing shape matching on each level. The position differences after a shape matching step are interpolated and added to the next finer level using our prolongation operator (line 9). During the prolongation it is important to propagate only the position differences from one level to another and not the actual positions, otherwise details get lost on finer levels.

Intergrid transfer operators The prolongation operator is a sparse block matrix $\mathbf{I}_l^{l+1} \in \mathbb{R}^{3n_f \times 3n_c}$ where n_c is the number of particles on the coarse level l and n_f is the number of particles on the fine level $l+1$.Georgii and Westermann [29] demonstrated that barycentric coordinates can be used to define a prolongation operator for unstructured and unrelated tetrahedral meshes. In order to propagate



(a) Multi-resolution model (b) Deformed model

Figure 4: (a) shows a multi-resolution model with three meshes. (b) The coarse meshes are compressed when bending the model. Therefore, shape matching on the coarse levels will influence the bending of the model.

information from level l to level $l+1$, we also use a linear interpolation based on barycentric coordinates. For each fine vertex i of level $l+1$ we first search the triangle on level l with the smallest distance. Then we project the fine vertex onto the plane of the triangle and compute its barycentric coordinates w_{ij}. Since $\sum_j w_{ij} = 1$ we can use these values directly as weights for our prolongation operator.

The restriction operator $\mathbf{I}_{l+1}^{l} \in \mathbb{R}^{3n_c \times 3n_f}$ is defined by transposing the prolongation operator and normalizing the row vectors of the resulting matrix. We need the normalization since we must guarantee that the summed up weight over all coarse particles is one.

Computational complexity When building the multi-resolution hierarchy for a mesh with n triangles, we reduce the number of triangles on each level by a constant factor $0 < r < 1$. Hence, the total number of triangles in a hierarchy with L levels can be described by the geometric series $\sum_{i=0}^{L} nr^i$. Since the absolute value of r is less than one, this series converges as L goes to infinity:

$$\sum_{i=0}^{\infty} nr^i = \frac{n}{1-r} \Leftrightarrow |r| < 1.$$

The computational complexity of shape matching is linear in the number of elements. Therefore, multi-resolution shape matching has a time complexity of $O(n)$.

Conserving fine wrinkles When using the multi-resolution approach as described above, the bending of the model can be influenced. Figure 4 shows that the coarse levels of a multi-resolution model are compressed if we have a fine wrinkle. Shape matching on the coarse levels as described above would try to eliminate this compression and so the fine details would get lost. In order to solve this problem we need a special handling of this compression case. Therefore, we introduce an adaption of the 1D and 2D shape matching methods presented in sections 4.2 and 4.1. This adaption is used on all levels of the hierarchy except the finest one.

Our method for conserving fine wrinkles is based on the same idea as the strain limiting approach of Wang et al. [16]. Both methods compute the principal strains and modify the deformation matrix. Wang et al. perform this modification for each triangle in a Gauss-Seidel iteration to limit the strain of their model globally. In contrast to them we only perform a special handling of the compression case on the coarse levels of our hierarchy in order to conserve fine wrinkles.

In order to handle a compressed coarse region without loosing the fine wrinkles, we have to take the scale of the region into account when computing the transformation for shape matching. A 1D shape matching region is compressed if the current length of the corresponding edge is smaller than its initial length. If this condition is fulfilled for a region on a coarse level, we simply skip 1D shape matching for this region. In this way fine wrinkles are not eliminated by a position change. Note, that Müller [24] proposed a similar approach for distance constraints in a hierarchical model.

In the case of 2D shape matching the handling of compressed regions is more difficult. In each shape matching step we determine the optimal rigid body transformation which consists of a translation vector and a rotation matrix. The optimal rotation matrix $\bar{\mathbf{R}}$ is the rotational part of $\bar{\mathbf{A}}_{pq}$ (see equation (10)) which we compute by a polar decomposition. However, since we are now also interested in the scale of a region, we have to take a look at the optimal linear transformation $\bar{\mathbf{A}} = \bar{\mathbf{A}}_{pq}\bar{\mathbf{A}}_{qq}^{-1}$ and not just the optimal rotation. Note, that the matrix $\bar{\mathbf{A}}_{qq}^{-1}$ can be precomputed for each region. The rotational part of the transformation $\bar{\mathbf{A}}$ is then determined by a polar decomposition $\bar{\mathbf{A}} = \bar{\mathbf{R}}_A\bar{\mathbf{V}}$. The matrix $\bar{\mathbf{V}} = \bar{\mathbf{R}}_A^T\bar{\mathbf{A}}$ is symmetric and positive definite and contains the shear and scale of the optimal transformation. The scale matrix $\bar{\mathbf{S}}$ can be determined by an eigendecomposition

$$\bar{\mathbf{V}} = \bar{\mathbf{W}}\bar{\mathbf{S}}\bar{\mathbf{W}}^T, \tag{16}$$

where $\bar{\mathbf{W}}$ is an orthogonal matrix which contains the eigenvectors of $\bar{\mathbf{V}}$ and $\bar{\mathbf{S}}$ is a diagonal matrix with the eigenvalues on the diagonal.

Now we know the scale of the region. If an eigenvalue is smaller than one, the region is compressed in the direction of the corresponding eigenvector. In this case we adapt the scale matrix in the following way:

$$\bar{\mathbf{S}}'_{ij} := \begin{cases} \bar{\mathbf{S}}_{ij} & \text{if } i = j \ \wedge \ \bar{\mathbf{S}}_{ij} \leq 1 \\ 1 & \text{if } i = j \ \wedge \ \bar{\mathbf{S}}_{ij} > 1 \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

If the region is stretched, we set the scale to one but in the case of a compression we do not change it. This means that goal positions will be compressed as well and we will not get a position change which eliminates fine details. An adapted transformation matrix is determined by

$$\bar{\mathbf{A}}' = \bar{\mathbf{R}}_A\bar{\mathbf{W}}\bar{\mathbf{S}}'\bar{\mathbf{W}}^T. \tag{18}$$

We use the transformation $\bar{\mathbf{A}}'$ instead of the rotation matrix in equation (12) in order to compute the goal positions for the region. For a region which is compressed in both

directions we do not have to evaluate equation (18) since $\bar{\mathbf{A}}'$ is equal to $\bar{\mathbf{A}}$ in this case. If we have no compression at all, we directly use the rotation matrix $\bar{\mathbf{R}}_A$ for shape matching. Note, that this matrix is the rotational part of the optimal linear transformation which can be slightly different from the optimal rotation matrix. However, since both matrices are almost equal, we prefer to use matrix $\bar{\mathbf{R}}_A$ instead of computing another polar decomposition for $\bar{\mathbf{A}}_{pq}$ to get the optimal rotation.

4.4. Additional Forces

Damping After each shape matching step we damp the velocities of the particles on the finest level of the multi-resolution hierarchy. Damping is performed per triangle to ensure that no out-of-plane forces are introduced. For each triangle we first compute its center of mass and its global linear velocity

$$\mathbf{x}_{cm} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i}, \quad \mathbf{v}_{cm} = \frac{\sum_i m_i \mathbf{v}_i}{\sum_i m_i}. \tag{19}$$

Then, the velocity change for each particle is determined by

$$\Delta \mathbf{v}_i = -k_d \left(\mathbf{r}_i^T \left(\mathbf{v}_i - \mathbf{v}_{cm} \right) \right) \mathbf{r}_i, \tag{20}$$

where k_d is the damping coefficient and $\mathbf{r}_i = \mathbf{x}_i - \mathbf{x}_{cm}$.

Bending The stretching and shearing stiffness must be much higher than the bending resistance of a realistic cloth model. Therefore, we must only handle small bending forces which we can add to the model and integrate with our explicit integration scheme (see section 3) without getting stability problems. We implemented the efficient quadratic bending model of Bergou et al. [30] which was developed for cloth models with high stretching stiffness and low bending resistance. However, we note that this term in the time integration does not guarantee unconditional stability anymore. This issue could be solved using a position-based bending model [23]. But since we never experienced stability problems, we preferred the quadratic bending model of Bergou et al. which yielded more compelling results.

5. Parallelization

The presented shape matching methods, the time integration as well as the computation of bending and damping forces are performed in parallel. We used OpenMP and CUDA [31] to implement the presented methods for multi-core CPUs and GPUs, respectively. The parallelization using these APIs are now described in more detail.

OpenMP In OpenMP a simple parallelization is employed which can be further optimized. The parallelization of the time integration with equation 2 is trivial. The restriction and prolongation operators (see section 4.3) can be represented as a sparse matrix. A simple parallel sparse matrix vector multiplication can be used to update the position change on different levels of the discretization. Damping and 2D shape matching work on a per triangle basis while 1D shape matching is computed per edge. The simulation of a bending constraint requires the four vertices of two adjacent triangles. Hence, we have groups of two, three and four vertices in our simulation. The corresponding particles need to be updated in a simulation step which generally leads to race conditions. Therefore, we determine independent sets for each vertex group type in a precomputation step in order to allow a parallel simulation. That means that each vertex in a set belongs to exactly one vertex group. Now, for each set the position, velocity and acceleration updates can be computed in parallel.

GPU parallelization As GPUs have a much higher degree of parallelization, a different strategy for some of the operations is employed. To achieve high performance it is furthermore very important to avoid memory transfers between CPU and GPU memory. We therefore perform all computations on the GPU.

The time integration is performed by evaluating equation (2). We use one thread per coordinate for the update. The restriction and prolongation operators (see section 4.3) as well as the matrix for the bending forces are constant and can be represented as sparse matrices. A simple parallel sparse matrix vector multiplication (SpMV) can be used to update the position change on different levels of the discretization or computing the bending forces, respectively. Both operations require a multiplication of one scalar value per entry of the sparse matrix with all three components of a vector. Therefore, we use the BIN sparse matrix data structure of Weber et al. [32] with a slight modification that adapts the SpMV operation. In order to efficiently use memory bandwidth, we only load one scalar entry per thread and multiply it with all three associated vector components. Other sparse matrix data structures (see e.g. the work of Bell et al. [33]) can be used to perform the SpMV operations as well. But without a modification the matrix entries must be tripled to match the dimension of the vectors.

The position updates for 1D and 2D shape matching are generally subject to race conditions since updates for edges and triangles with common vertices depend on each other. The OpenMP parallelization strategy for the position updates is not suited for a massive parallel architecture. Each independent set would require a separate kernel call for updating vertices. The overhead for launching a single kernel on a GPU is not negligible so the number of launches should be minimized [32]. The number of threads that can be used to process the vertices in parallel corresponds to the number of elements in the sets. E.g., for 1D shape matching on a regular triangle mesh (where all interior vertices are adjacent to six edges) six independent sets are needed and only a sixth of all operations can be processed in parallel. This gets even worse for non-regular meshes or when applying 2D shape matching. We therefore propose a strategy that reduces the number kernel

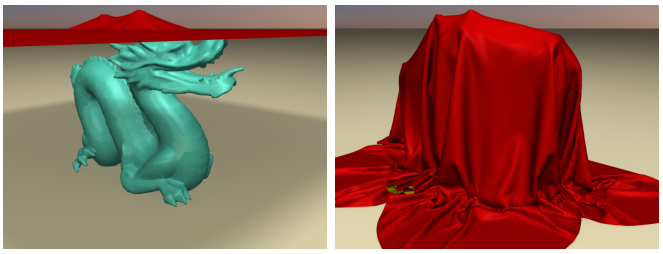


Figure 5: Simulation of a quadratic piece of cloth falling over a dragon. The finest level mesh consists of 124017 triangles. The simulation with five hierarchy levels requires 11.47 ms per step on average to compute the deformation on the GPU.

calls and additionally avoids a low utilization of the GPU. In contrast to the OpenMP version, we compute the goal positions per element (edge or triangle) in a first step and store the results locally per triangle and edge without updating the final vertex positions. In a second step shape matching is then completed by collecting and summing up the contributions of all elements adjacent to a vertex. A similar subdivision in a per element and a per vertex pass was also used by Allard et al. [34] to implement an implicit FEM solver on the GPU. The edges and triangles that are adjacent to a vertex are determined in a pre-processing step and stored in lists. There is one list per vertex that is processed by one thread to update the global state of the particles.

6. Results

The simulations presented in this section were performed on two Intel X5650 processors with 2.66 GHz and a GeForce GTX 470. For all simulations we used the stiffness coefficients $\alpha = 1$ and $\beta = 1$ since we want to simulate stiff materials. We generated the multi-resolution hierarchies for the models by using a half-edge collapse operator [28] to decimate the corresponding meshes. Approximately 50 % of the triangles were removed per level. We treated collisions and contact with friction by using the robust collision handling method of Bridson et al. [17]. Since collision handling is not the focus of our paper, the measured computation times in this section do not include collision detection and collision response.

Performance We first simulated a quadratic piece of cloth with different mesh resolutions in order to show the scalability of our method. Meshes between 10000 and 100000 triangles were generated for the simulation. We used a nested model and computed five hierarchy levels for each of these meshes. Figure 6 shows the average computation times required for one simulation step. Obviously, shape matching, time integration as well as the computation of bending and damping forces are performed in linear time. The computational costs of all steps are shown in table 2.

To show the power of our GPU implementation we generated another cloth model with 206664 triangles and five

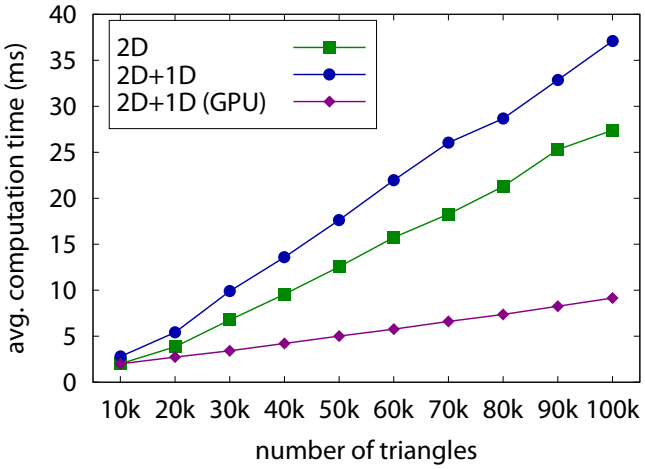


Figure 6: Average computation times of a simulation step with 2D shape matching and the combination of 1D and 2D shape matching for different model sizes.

hierarchy levels. For this model only 25 % of the triangles were removed per level which results in a more complex model with 630480 triangles on all five levels. The timings for this model as well as for the models of figures 1 and 5 are listed in table 1. The results show that the speed-up factor of our parallel implementations depends on the size of the model. For the largest model with more than 200 thousand triangles we measured a speed-up factor of 7.1 on 12 cores in comparison to a single core. The factor of the GPU implementation was 33.2 and the simulation of the deformation required only 22.58 ms.

Most previous works use spring-mass systems or a continuous model in combination with the finite element method to simulate cloth. Therefore, in our last performance experiment we compare our multi-resolution shape matching approach with a spring-mass simulation and a finite element method. All simulations were performed in parallel on the CPU and use the same bending model to get a fair comparison. For the comparison we use the cloth model in figure 1. We adjusted the stiffness parameters of the spring-mass and the finite element method to get a comparable average strain in all three simulations. The spring-mass method uses a simple cloth model with a particle at each vertex and a spring on each edge in combination with an implicit Euler integrator. Note that shearing resistance is not directly simulated by this method. To solve the system of linear equations in the integration step, we use an highly optimized, parallel preconditioned conjugate gradient method. Even though this is a simple and fast simulation method, multi-resolution shape matching was about 9 times faster than the spring-mass simulation. The second method, we used for our comparison, is the fast corotational finite element method for arbitrary triangle meshes introduced by Etzmuss et al. [10]. Compared to this method multi-resolution shape matching was more than 15 times faster. In conclusion, our method provides

Model	triangles (l_{max})	triangles (total)	single core	multi-core	speed-up	GPU	speed-up
Dancing Woody	32467	62888	61.81 ms	11.63 ms	5.3	3.89 ms	15.9
Dragon	124017	240275	267.33 ms	46.15 ms	5.8	11.47 ms	23.3
Large cloth	206664	630480	749.83 ms	105.99 ms	7.1	22.58 ms	33.2

Table 1: Timings of three different models with five hierarchy levels on a single core, on multiple cores and on a GPU. On the multi-core processor and the GPU the time integration, the computation of the bending and damping forces as well as both shape matching methods were performed in parallel. The table contains the measured timings and the speed-up of our multi-core and GPU implementations in comparison to the single core implementation.

Model	integr.	bend.	damp.	SM 2D	SM 1D
Dragon	1.4 %	13.3 %	7.2 %	53.5 %	24.6 %
Woody	1.4 %	13.5 %	5.8 %	52.9 %	26.4 %
Large cl.	0.5 %	9.0 %	4.1 %	56.6 %	29.8 %

Table 2: Computational costs of the different steps in percent. The two shape matching steps require about 80 percent of the total computation time.

a significant speed-up compared to traditional methods at the cost of accuracy.

Stiffness In order to show how the strain of a model depends on the number of hierarchy levels, we simulated a quadratic cloth model with 20648 triangles using different numbers of nested hierarchy levels. The model had a size of 5 m×5 m and was fixed at two corners. It was stretched under its own weight due to gravity. We measured the average local strain of the model and the maximum global strain. The results for a time step size of 5 ms are shown in figure 7. The average strain is reduced by a factor of approximately 1.7 when using our combination of 1D and 2D shape matching instead of 2D shape matching only. Since 1D shape matching is very simple and about two times faster than the 2D method (see table 2), 1D shape matching is an important extension for our 2D shape matching. Using the combined methods we measured an average local strain of less than 5 % with only three hierarchy levels. For a global maximum strain of 5 % we needed seven levels.

The computation time which is required for a simulation step does not depend on the time step size but the stiffness of the model does. Therefore, we carried out a second experiment with the same cloth model but this time we used a fixed number of five levels and changed the time step size. In table 3 we can see that we have an average strain of less than 10 % for the combination of 1D and 2D shape matching with a time step size of 10 ms. Using a size of 2.5 ms the average strain was less than 1 % in both cases. The global maximum strain for our combined method was 8 % for a time step size of 5 ms and less than 3 % for a step size of 2.5 ms. The table shows that the strain depends almost quadratically on the time step size.

Stability In order to test the stability of our method, we moved the vertices of a cloth model with 20000 triangles and five non-nested hierarchy levels to random positions before starting the simulation (see figure 8). After a few simulation steps the multi-resolution shape matching approach was able to recover the model from the invalid state.

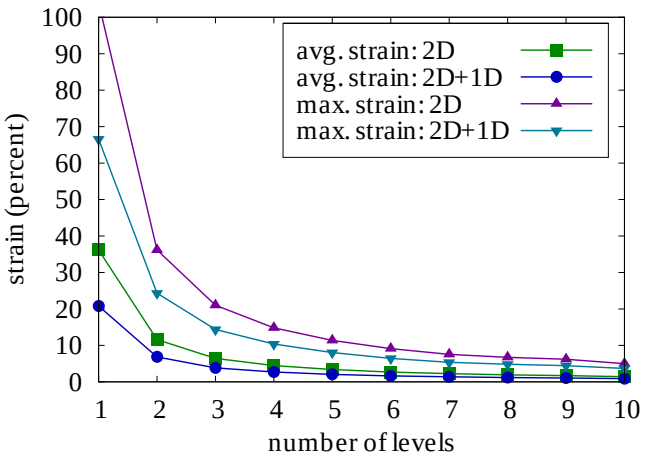


Figure 7: Average local strain and maximum global strain of a model with 20648 triangles and a different number of hierarchy levels using a time step size of 5 ms.

strain	20 ms	10 ms	5 ms	2.5 ms	1.25 ms
avg.: 2D	56.41	13.55	3.39	0.85	0.21
avg.: 2D+1D	34.44	8.29	2.06	0.52	0.13
max.: 2D	142.84	39.59	11.37	3.29	1.12
max.: 2D+1D	96.85	27.64	8.03	2.28	0.74

Table 3: Average local strain and maximum global strain (percent) of a cloth model with 20648 triangles and five hierarchy levels for different time step sizes h.

We ran another test where we collapsed all vertices into a single point. All triangles in this test were degenerated in the initial configuration. Hence, their projection matrices (see equation (7)) were not uniquely defined. Therefore, we simply projected the triangles in the xz-plane in the first simulation step and then continued as described before. Afterwards our approach was even able to handle this extremely degenerated configuration.

Conservation of fine wrinkles The last simulations demonstrate the benefit of our method to conserve fine wrinkles which was introduced in section 4.3. Figure 9 shows a piece of cloth with 20000 triangles and four nested hierarchy levels falling over a sphere. Thanks to our wrinkle conservation approach we can see significantly more fine details in the right image than in the left one.

Another experiment was made simulating a cloth model with 30000 triangles and four non-nested hierarchy levels without gravity (see figure 10). The y-coordinate

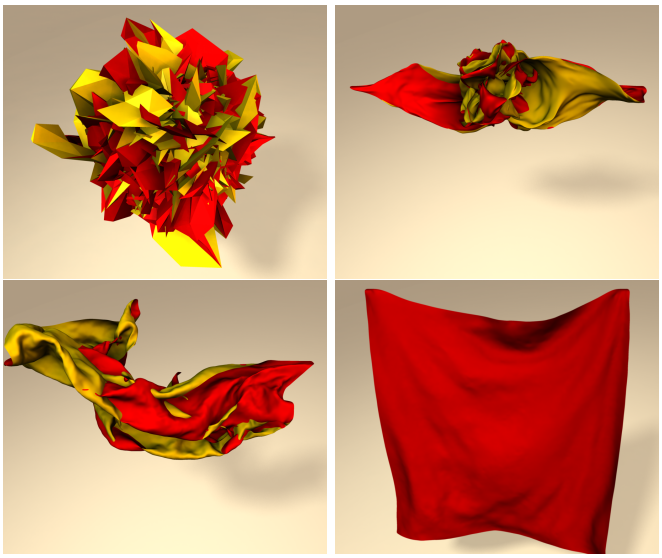


Figure 8: Stability test. The vertices of a cloth model are moved to random positions before the simulation starts.

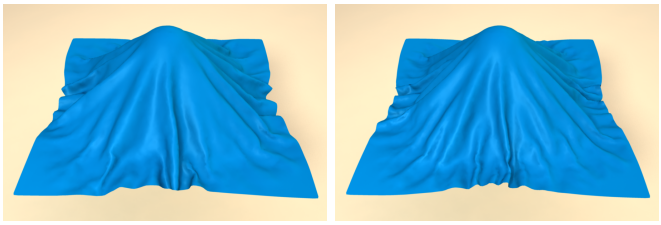


Figure 9: Cloth with 20000 triangles and four hierarchy levels falling over a sphere. The coarse levels of the multi-resolution model prevent the appearance of fine wrinkles (left). Our method to conserve fine wrinkles solves this problem (right).

of the center particle was animated during the simulation using a sine curve. Hence, it moves up and down and causes a wave on the cloth. The simulation was performed once with our method to conserve fine wrinkles (see figure 10(b)) and once without it (see figure 10(c)). The result of a simulation using the corotational finite element method introduced by Etzmuss et al. [10] serves as reference (see figure 10(a)). Here we adjusted the stiffness parameters to get a comparable average strain. Since shape matching is geometrically motivated, a direct accuracy comparison to a finite element simulation makes no sense. However, if we compare the results visually, we see that our shape matching method provides physically plausible results. Moreover, we can also see that thanks to our wrinkle conservation method the fine details in figure 10(b) are similar to the ones in 10(a). Without applying this method fine wrinkles get lost as we see in figure 10(c).

Discussion Multi-resolution shape matching has several advantages. It is simple to implement, is unconditionally stable and has no numerical damping. The multi-resolution model allows for a high stretching and shearing resistance which is required for the simulation of realistic fabrics. Since a model is simulated using multiple meshes with different resolutions, forces are propagated faster through the mesh which is another advantage of the multi-resolution approach. Furthermore, the proposed method is very fast and simulations are performed in linear time. The results show that the deformations of a cloth model with more than 200 thousand triangles and five hierarchy levels can be simulated at interactive frame rates on a GPU.

We also want to discuss the drawbacks of our methods. Since multi-resolution shape matching is a geometric approach, only physically plausible results can be obtained. Similar to all other shape matching approaches, the stiffness of the model does not only depend on the stiffness coefficient but also on the time step size as well as the region sizes or, in our case, the number of hierarchy levels. Furthermore, the physical behavior depends on the mesh of the model. Therefore, adaptive time stepping and level-of-detail methods cannot be used without influencing the behavior of the model.

7. Conclusion

We presented a novel simple, efficient and unconditionally stable cloth simulation method based on shape matching. In order to enlarge the influence of the shape matching regions we developed a multi-resolution approach. This allows for simulating stiff materials and therefore we get more realistic results. We introduced a simple parallelization of our method for multi-core CPUs by using independent sets of vertex groups and an efficient GPU implementation.

At the moment the bottleneck of the simulation is the collision detection and resolution. Therefore, we plan to implement a GPU-based collision detection in order to transfer the whole simulation pipeline to the GPU. On the other hand, we want to perform the collision handling on a coarser level of the multi-resolution hierarchy since we do not always need the high resolution of the finest mesh for a plausible collision handling. If the collision handling is performed on a coarse level, we must prevent the introduction of new interpenetrations during the prolongation phase. This should be achieved by adjusting the weights of the resolved particles during shape matching and prolongation.

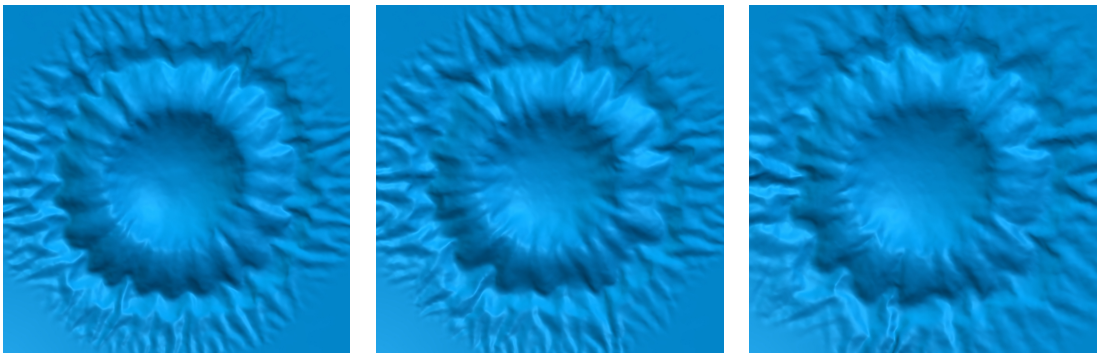
References

[1] Müller M, Heidelberger B, Teschner M, Gross M. Meshless deformations based on shape matching. ACM Trans Graph 2005;24(3):471–8.

[2] Magnenat-Thalmann N, Volino P. From early draping to haute couture models: 20 years of research. The Visual Computer 2005;21:506–19.

[3] Nealen A, Müller M, Keiser R, Boxerman E, Carlson M. Physically based deformable models in computer graphics. Computer Graphics Forum 2006;25(4):809–36.

[4] Breen DE, House DH, Wozny MJ. Predicting the drape of woven cloth using interacting particles. In: Proc. of SIGGRAPH 1994. New York, NY, USA: ACM; 1994, p. 365–72.



(a) FEM simulation (b) Conserving fine wrinkles (c) Without conserving fine wrinkles

Figure 10: Top view of a cloth model with 30000 triangles and four hierarchy levels. We animated the y-coordinate of the center particle using a sine curve in order to cause a wave motion. The results show that our wrinkle conservation method preserves fine details (b) which are similar to the one of a finite element simulation (a) while these details get lost without using our method (c).

[5] Provot X. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In: In Graphics Interface. Canadian Human-Computer Communications Society; 1995, p. 147–54.

[6] Hauth M, Etzmuß O, Straßer W. Analysis of numerical methods for the simulation of deformable models. The Visual Computer 2003;19(7-8):581–600.

[7] Baraff D, Witkin A. Large steps in cloth simulation. In: Proc. of SIGGRAPH 1998. New York, NY, USA: ACM; 1998, p. 43–54.

[8] Choi KJ, Ko HS. Stable but responsive cloth. ACM Trans Graph 2002;21(3):604–11.

[9] Bridson R, Marino S, Fedkiw R. Simulation of clothing with folds and wrinkles. In: Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association; 2003, p. 28–36.

[10] Etzmuss O, Keckeisen M, Strasser W. A fast finite element solution for cloth modelling. In: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications. PG '03; Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-2028-6; 2003, p. 244.

[11] Volino P, Magnenat-Thalmann N, Faure F. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. ACM Trans Graph 2009;28(4):105:1–105:16.

[12] Goldenthal R, Harmon D, Fattal R, Bercovier M, Grinspun E. Efficient simulation of inextensible cloth. ACM Trans Graph 2007;26(3).

[13] Bender J, Bayer D. Parallel simulation of inextensible cloth. In: Proc. of Virtual Reality Interactions and Physical Simulations. 2008, p. 47–56.

[14] English E, Bridson R. Animating developable surfaces using nonconforming elements. ACM Trans Graph 2008;27(3).

[15] Thomaszewski B, Pabst S, Strasser W. Continuum-based Strain Limiting. Computer Graphics Forum 2009;28(2):569–76.

[16] Wang H, O'Brien J, Ramamoorthi R. Multi-resolution isotropic strain limiting. ACM Trans Graph 2010;29(6):156:1–156:10.

[17] Bridson R, Fedkiw R, Anderson J. Robust treatment of collisions, contact and friction for cloth animation. ACM Trans Graph 2002;21:594–603.

[18] Bender J, Müller M, Otaduy MA, Teschner M. Position-based methods for the simulation of solid objects in computer graphics. In: EUROGRAPHICS 2013 State of the Art Reports. Eurographics Association; 2013, p. 1–22.

[19] Rivers AR, James DL. FastLSM: Fast lattice shape matching for robust real-time deformation. ACM Trans Graph 2007;26(3).

[20] Steinemann D, Otaduy MA, Gross M. Fast adaptive shape matching deformations. In: Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association; 2008, p. 87–94.

[21] Diziol R, Bender J, Bayer D. Robust real-time deformation of incompressible surface meshes. In: Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation. New York, NY, USA: ACM; 2011, p. 237–46.

[22] Müller M, Chentanez N. Solid simulation with oriented particles. ACM Trans Graph 2011;30(4):92:1–92:10.

[23] Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. In: Proc. of Virtual Reality Interactions and Physical Simulations. 2006, p. 71–80.

[24] Müller M. Hierarchical Position Based Dynamics. In: Proc. of Virtual Reality Interactions and Physical Simulations. Eurographics Association; 2008, p. 1–10.

[25] Stumpp T, Spillmann J, Becker M, Teschner M. A Geometric Deformation Model for Stable Cloth Simulation. In: Proc. of Virtual Reality Interactions and Physical Simulations. 2008, p. 39–46.

[26] Shoemake K, Duff T. Matrix animation and polar decomposition. In: Proc. of Graphics interface. Morgan Kaufmann Publishers Inc.; 1992, p. 258–64.

[27] Dick C, Georgii J, Westermann R. A hexahedral multigrid approach for simulating cuts in deformable objects. IEEE TVCG 2011;17(11):1663–75.

[28] Kobbelt L, Campagna S, Seidel HP. A general framework for mesh decimation. In: Graphics Interface. 1998, p. 43–50.

[29] Georgii J, Westermann R. A multigrid framework for real-time simulation of deformable bodies. Computer & Graphics 2006;30:408–15.

[30] Bergou M, Wardetzky M, Harmon D, Zorin D, Grinspun E. A quadratic bending model for inextensible surfaces. In: Proc. of the fourth Eurographics symposium on Geometry processing. SGP '06; Eurographics Association; 2006, p. 227–30.

[31] NVIDIA . NVIDIA CUDA Compute Unified Device Architecture - Programming Guide; 2012. Version 4.1, http://nvidia.com/cuda.

[32] Weber D, Bender J, Schnoes M, Stork A, Fellner D. Efficient GPU data structures and methods to solve sparse linear systems in dynamics applications. Computer Graphics Forum 2013;32(1):16–26.

[33] Bell N, Garland M. Efficient sparse matrix-vector multiplication on CUDA. NVIDIA Technical Report NVR-2008-004; NVIDIA Corporation; 2008.

[34] Allard J, Courtecuisse H, Faure F. Implicit FEM and fluid coupling on GPU for interactive multiphysics simulation. In: SIGGRAPH Talks. ACM; 2011, p. 52–.