Computers & Graphics 37 (2013) 820–829

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Adaptive cloth simulation using corotational finite elements[☆]

Jan Bender[*], Crispin Deul

Graduate School CE, TU Darmstadt, Germany

A R T I C L E I N F O

Article history:
Received 1 February 2013
Received in revised form
15 April 2013
Accepted 30 April 2013
Available online 20 May 2013

Keywords:
Cloth simulation
Finite element method
Adaptive model

A B S T R A C T

In this article we introduce an efficient adaptive cloth simulation method which is based on a reversible $\sqrt{3}-$refinement of corotational finite elements. Our novel approach can handle arbitrary triangle meshes and is not restricted to regular grid meshes which are required by other adaptive methods. Most previous works in the area of adaptive cloth simulation use discrete cloth models like mass-spring systems in combination with a specific subdivision scheme. However, if discrete models are used, the simulation does not converge to the correct solution as the mesh is refined. Therefore, we introduce a cloth model which is based on continuum mechanics since continuous models do not have this problem. We use a linear elasticity model in combination with a corotational formulation to achieve a high performance. Furthermore, we present an efficient method to update the sparse matrix structure after a refinement or coarsening step.

The advantage of the $\sqrt{3}-$subdivision scheme is that it generates high quality meshes while the number of triangles increases only by a factor of 3 in each refinement step. However, the original scheme was not intended for the use in an interactive simulation and only defines a mesh refinement. In this article we introduce a combination of the original refinement scheme with a novel coarsening method to realize an adaptive cloth simulation with high quality meshes. The proposed approach allows an efficient mesh adaption and therefore does not cause much overhead. We demonstrate the significant performance gain which can be achieved with our adaptive simulation method in several experiments including a complex garment simulation.

1. Introduction

Interactive cloth simulation has a long history in computer graphics. In this area the resolution of the simulation mesh plays an important role. On one hand the resolution must be high enough to get realistic wrinkles during the simulation, on the other hand simulations with high detailed meshes cost much computation time and often do not run at interactive frame rates. In this article we present a cloth simulation method which changes the resolution of the cloth model adaptively. In regions of the model with fine wrinkles small triangles are used for the simulation while a low resolution is used in areas without fine details. The advantage of such an adaptive model is that the performance can be increased significantly without loosing much details.

The idea of using an adaptive mesh as cloth model is not new. There exist different works which focus on this topic. Most of the previous approaches use adaptive mass-spring systems for the simulation. In general such systems are not convergent, i.e. the simulation does not converge to the correct solution as the mesh is refined [1]. To solve this problem we introduce an adaptive cloth model based on continuum mechanics. We use a linear finite element method (FEM) in combination with a corotational formulation to perform the simulation efficiently. This method works on triangular elements which are defined by the adaptive triangle mesh of our cloth model. The resolution of this mesh is adapted during the simulation by using a $\sqrt{3}-$subdivision scheme [2]. This scheme defines how a triangle mesh can be refined adaptively while maintaining a high mesh quality. In this article we present an extension which allows us to coarsen the mesh in areas where a fine resolution is not required anymore.

In contrast to other adaptive simulation methods, our approach can handle arbitrary triangle meshes and is not restricted to meshes based on regular grids. Our refinement criterion is based on the mean curvature. Therefore, we get a high resolution for fine wrinkles and a low resolution in flat regions. The proposed method can speed up the simulation significantly at the cost of accuracy. The performance gain and therefore also the accuracy loss can be controlled indirectly by the user-defined parameters of the refinement criterion. The mesh adaption with our method can be performed very efficiently. Hence, the computational overhead caused by the adaption is low.

This article is an extended version of our earlier paper [3]. In comparison to our earlier paper we added the description of

[☆] This article was recommended for publication by Dr M. Teschner.
[*] Corresponding author.
E-mail address: bender@gsc.tu-darmstadt.de (J. Bender).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.04.008

a hybrid method to determine the area in the triangle model which is represented by a single particle. These particle areas are required for a consistent mass distribution and to compute the mean curvature of the model. The mean curvature is needed by the refinement criterion in our adaptive remeshing process. We also added more details about the refinement process as well as the computation of the bending matrix and the damping matrix. Furthermore, we developed a fast matrix assembly strategy. Since we use an adaptive cloth model, the matrix structure changes after each refinement and coarsening step. The presented matrix assembly allows us to update the structure efficiently. As a final extension we present new experiments to demonstrate the contribution of our simulation method in a complex garment simulation.

2. Related work

In this section we want to give an overview over important works in the area of cloth simulation and adaptive deformable models.

Research in cloth simulation has been done for more than 20 years in the field of computer graphics (for surveys see [4,5]). Often the assumption is made that cloth is an elastic material in order to perform an efficient simulation using spring forces. The problem is that many real textiles cannot be stretched significantly. Different techniques have been presented to solve this problem. Provot [6] used a mass-spring system for cloth simulation in combination with an explicit time integration. Instead of using stiff springs which can cause instabilities, Provot proposed to displace particle positions after each simulation step as an alternative way for strain reduction. Baraff and Witkin [7] used an implicit Euler integration in order to perform a stable simulation for stiff systems. This approach supports the simulation of arbitrary triangle meshes whereas other approaches require regular grid structures, e.g. [6,8]. A semi-implicit method was used by Choi and Ko [8] for a stable simulation with stiff springs. They also solved the problem with instabilities of the post-buckling response which are not caused by stiff equations. In order to limit the strain, Bridson et al. [9] applied corrective impulses to the velocities of the particles. Goldenthal et al. [10] presented an approach based on Lagrangian mechanics in combination with a fast projection method in order to simulate inextensible cloth. English and Bridson [11] performed cloth simulations using triangle meshes with a hard constraint on each edge. In order to solve the consequential locking problem, they used a nonconforming mesh for the simulation which has more degrees of freedom than the original one. Bender et al. [12] combined this technique with an impulse-based approach [13] to simulate models with hard constraints more efficiently. A continuum-based strain limiting method was introduced by Thomaszewski et al. [14].

In the last years different authors proposed to use continuous models to simulate cloth. In contrast to discrete models like mass-spring systems, a model based on continuum mechanics has the advantage that it is independent of the mesh resolution. Etzmuss et al. [15] used a finite difference discretization of the model in order to solve the differential equations. Due to this discretization only quadrilateral meshes can be handled. In a second work they presented an efficient approach based on the finite element method (FEM) with a corotational formulation which can also handle arbitrary triangle meshes [16]. Thomaszewski et al. [17] also use a corotational formulation for their finite element simulation. In their work they show how membrane and bending energies can be modeled consistently for thin, flexible objects. Volino et al. [18] present a cloth simulation system based on continuum mechanics which is able to simulate nonlinear anisotropic materials.

There exist different approaches to improve the performance of cloth simulations by using an adaptive refinement of the simulation model. Hutchinson et al. [19] presented an adaptive mass-spring model for cloth simulation. This model has a regular grid structure which is refined when the angle between two neighboring springs exceeds a certain tolerance value. A similar approach which also uses regular quad meshes in combination with a mass-spring model was introduced in [20]. Li and Volkov [21] presented an adaptive version of Baraff's cloth simulation method [7] which is able to handle arbitrary triangle meshes. They use a modified $\sqrt{3}-$refinement rule without explicit edge flip which forces a subdivision of adjacent triangles. Hence, the number of triangles increases faster compared to our method. Lee et al. [22] use a mass-spring system in combination with a Loop subdivision scheme for refining a triangle model. The subdivision steps are precomputed in order to get a multi-resolution hierarchy. This is used to adaptively reduce the dimension of the linear system which must be solved for an implicit integration step. In contrast to these previous works that use mass-spring systems which are not convergent, our model is based on continuum mechanics. Brochu et al. [23] use the continuous cloth model proposed by Etzmuss et al. [15] and perform simple edge splitting, flipping and collapsing in order to demonstrate that their continuous collision detection is able to handle adaptive meshes. Grinspun et al. [24] use a continuous model for the adaptive simulation of thin shells and volumetric deformable models. But instead of refining the elements, they introduce a refinement of the basis functions to reduce the computation time of a simulation step. Further adaptive methods for volumetric deformable models are presented in [25,26].

3. Simulation step

This section gives a short overview over the time integration of the adaptive cloth simulation method. In the following sections each step will be explained in detail.

For the simulation we use a triangular mesh of particles as cloth model. Each particle has a mass m, a position $\mathbf{x}$ and a velocity $\mathbf{v}$. A single simulation step is performed as follows:

1. Determine all external forces which are acting on the model.
2. Perform a simulation step with the continuous model to get new positions $\mathbf{x}^{n+1}$ (see Section 4).
3. Determine average velocities $\mathbf{v}^{n+1/2} = (\mathbf{x}^{n+1} - \mathbf{x}^n)/\Delta t$.
4. Detect proximities for $\mathbf{x}^n$ and resolve them with friction by modifying the average velocities $\mathbf{v}^{n+1/2}$ with impulses (see Section 6).
5. Perform a continuous collision detection step for the linear trajectory from $\mathbf{x}^n$ to $\mathbf{x}^n + \Delta t \mathbf{v}^{n+1/2}$ and adapt the average velocities $\mathbf{v}^{n+1/2}$ by applying impulses to resolve collisions with friction (see Section 6).
6. Compute final positions and velocities (see Section 6).
7. Adapt the resolution of the mesh (see Section 5).

4. Cloth simulation

In this section we first introduce our cloth simulation model. Then we introduce the corotational formulation for linear elasticity in order to simulate stretching and shearing. Furthermore, we show how the simulation of bending and damping is realized. In the end of this section we briefly introduce an implicit time integration method which is used to simulate stiff fabrics without stability problems.

4.1. Cloth model

Our cloth model is based on continuum mechanics and we use an arbitrary triangle mesh to define elements for solving the equation of motion with the finite element method.

The mass distribution of our cloth model is defined by a diagonal mass matrix **M**. In this way the masses of the model are concentrated at the vertices of the mesh. We assume that the simulated material has a homogeneous mass distribution and therefore a constant density. Furthermore, we assume that the mass of a dynamic particle is proportional to the area of the triangles adjacent to its corresponding vertex in the triangle mesh. The area of this vertex can be defined by the Voronoi area which is bounded by the midpoints of the incident edges and the circumcenters of the adjacent triangles (see Fig. 1(a)). This works well for triangles which are not obtuse. For a non-obtuse triangle t with the vertices $\mathbf{x}_i$, $\mathbf{x}_j$ and $\mathbf{x}_k$ the Voronoi area of a vertex is determined by

$$A_{\text{Voronoi}}(t,\mathbf{x}_i)=\frac{1}{8}(\|\mathbf{x}_k-\mathbf{x}_i\|^2\cot\varphi_j+\|\mathbf{x}_j-\mathbf{x}_i\|^2\cot\varphi_k), \qquad (1)$$

where φ_i denotes the angle in the triangle at the vertex $\mathbf{x}_i$. If all triangles adjacent to a vertex $\mathbf{x}_i$ are non-obtuse, the Voronoi area of this vertex is

$$A_{\text{Voronoi}}(\mathbf{x}_i)=\frac{1}{8}\sum_{j\in R_1(i)}(\cot\alpha_{ij}+\cot\beta_{ij})\|\mathbf{x}_i-\mathbf{x}_j\|^2, \qquad (2)$$

where the set $R_1(i)$ defines the one-ring of vertex $\mathbf{x}_i$. α_{ij} and β_{ij} are the opposite angles of the edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ (see Fig. 1(b)).

Obtuse triangles need a special treatment. The area computed for the acute angled vertices in such a triangle is negative if the total area of all vertices should equal the area of the triangle. This problem can be solved by the hybrid approach of Meyer et al. [27] which is implemented in Algorithm 1.

Algorithm 1.

```
1: $A_{\text{hybrid}}(\mathbf{x})=0$
2: for all triangles $t$ in the one-ring of $\mathbf{x}$ do
3:   if If $t$ is not obtuse then
4:     $A_{\text{hybrid}}(\mathbf{x})+=A_{\text{Voronoi}}(t,\mathbf{x})$
5:   else
6:     Compute area $A_t$ of triangle $t$
7:     if If angle at $\mathbf{x}$ in $t$ is obtuse then
8:       $A_{\text{hybrid}}(\mathbf{x})+=\frac{1}{2}A_t$
9:     else
10:      $A_{\text{hybrid}}(\mathbf{x})+=\frac{1}{4}A_t$
11:    end if
12:  end if
13: end for
```

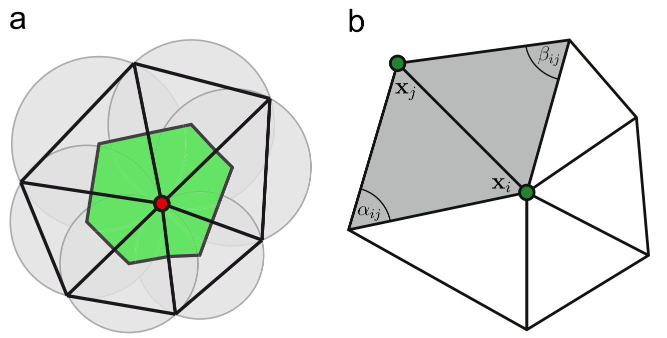


Fig. 1. (a) The Voronoi area of a vertex in a triangle mesh. This area is bounded by the midpoints of the incident edges and the circumcenters of the adjacent triangles. (b) The one-ring of the vertex $\mathbf{x}_i$ and the opposite angles α_{ij} and β_{ij} of an edge.

The final mass of a particle m is determined by the product of its corresponding area $A_{\text{hybrid}}(\mathbf{x})$ with a user-defined factor ρ in order to account for the density of the simulated object:

$$m=A_{\text{hybrid}}(\mathbf{x})\cdot\rho. \qquad (3)$$

In each simulation step with our adaptive model the number of particles changes. Therefore, we have to adapt the masses of the particles in order to guarantee the mass conservation of the model. This mass adaption has to be performed after each refinement and coarsening step but only for the particles where the adjacent triangles changed.

4.2. Simulation

In this work we use a finite element discretization of our continuous model in order to perform a simulation step. Using the Lagrange form we get the following ordinary differential equations which describe the dynamics of our model:

$$\mathbf{M}\ddot{\mathbf{x}}+\mathbf{D}\dot{x}+\mathbf{K}(\mathbf{x}-\mathbf{x}_0)=\mathbf{f}_{\text{ext}}, \qquad (4)$$

where **M** is the mass matrix with the masses of the particles on the diagonal, **D** is a damping matrix and **K** is the stiffness matrix of the cloth model. The vectors $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ contain the positions, velocities and accelerations, respectively, at the vertices of the simulation mesh while $\mathbf{x}_0$ defines the rest state of the model. Hence, we have $3n$ differential equations for a mesh with n vertices.

4.2.1. Linear elasticity

The simulation of the stretching and shearing behavior of our cloth model can be performed in the two-dimensional space of the triangle mesh. Therefore, we define the deformation of our model by a two-dimensional vector field $\mathbf{u}(\mathbf{m})$ which is used to compute the deformed point location $\mathbf{x}(\mathbf{m})=\mathbf{m}+\mathbf{u}(\mathbf{m})$ of an undeformed point $\mathbf{m}\in\mathbb{R}^2$. This vector field is only defined in areas where material exists.

To perform a finite element simulation the continuous displacement function $\mathbf{u}(\mathbf{m})$ is evaluated only at the vertices of our triangular simulation mesh. The displacement in the interior of each triangular element is interpolated linearly by three shape functions $N_i(\mathbf{m})$:

$$\mathbf{u}(\mathbf{m})=\sum_{i=1}^{3}N_i(\mathbf{m})\cdot\hat{\mathbf{u}}_i. \qquad (5)$$

The vectors $\hat{\mathbf{u}}_i\in\mathbb{R}^2$ contain the displacements at the vertices of the element. In Appendix A we describe how the shape functions and the corresponding derivatives are determined.

For our simulation we use Cauchy's linear strain tensor

$$\epsilon=\frac{1}{2}\left(\frac{\partial\mathbf{u}}{\partial\mathbf{m}}+\frac{\partial\mathbf{u}}{\partial\mathbf{m}}^T\right), \qquad (6)$$

which is in our case a symmetric 2×2 tensor. Therefore, it can also be written as vector with three entries:

$$\begin{pmatrix}\epsilon_x\\ \epsilon_y\\ \gamma_{xy}\end{pmatrix}=\sum_{i=1}^{3}\mathbf{B_i}\hat{\mathbf{u}}_i \quad\text{with}\quad \mathbf{B_i}=\begin{pmatrix}\frac{\partial N_i}{\partial x} & 0\\ 0 & \frac{\partial N_i}{\partial y}\\ \frac{1}{2}\frac{\partial N_i}{\partial y} & \frac{1}{2}\frac{\partial N_i}{\partial x}\end{pmatrix}. \qquad (7)$$

Following Hooke's law, we can describe the stress within an element by the tensor

$$\sigma = \mathbf{C}\epsilon = \mathbf{C}\mathbf{B}_e\hat{\mathbf{u}}, \tag{8}$$

where $\mathbf{B}_e = (\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3)$ is a 3×6 matrix and $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_1^T, \hat{\mathbf{u}}_2^T, \hat{\mathbf{u}}_3^T)^T$ is a six-dimensional vector. **C** is a tensor which describes the elasticity of the material. Woven textiles have a weft and a warp direction. Therefore, we model cloth as orthotropic material with two orthogonal symmetry axes. The elasticity tensor is defined by two Young moduli E_x and E_y for the weft and the warp direction by a shear modulus E_s and the Poisson's ratios ν_{xy} and ν_{yx}:

$$\mathbf{C} = \begin{pmatrix} \frac{E_x}{1-\nu_{xy}\nu_{yx}} & \frac{E_x\nu_{yx}}{1-\nu_{xy}\nu_{yx}} & 0 \\ \frac{E_y\nu_{xy}}{1-\nu_{xy}\nu_{yx}} & \frac{E_y}{1-\nu_{xy}\nu_{yx}} & 0 \\ 0 & 0 & E_s \end{pmatrix}. \tag{9}$$

Poisson's ratio ν_{xy} corresponds to a contradiction in direction y when the material is extended in direction x.

Now we can compute the forces which are acting on the three vertices of a triangular element:

$$\mathbf{f}_e = A_e\mathbf{B}_e^T\mathbf{C}\mathbf{B}_e\hat{\mathbf{u}} = \mathbf{K}_e\hat{\mathbf{u}}, \tag{10}$$

where A_e is the initial area of the corresponding triangle, $\mathbf{K}_e \in \mathbb{R}^{6\times 6}$ is the stiffness matrix of the element and the vector $\overline{\mathbf{f}}_e \in \mathbb{R}^6$ contains the forces for the three vertices in material coordinates.

4.2.2. *Corotational formulation*

A stable simulation with a linear elasticity model can be performed efficiently with an implicit integration scheme. However, such a model is not suitable for large rotational deformations since nonlinear effects can cause undesired deformations [28]. To solve this problem we use a corotational formulation similar to the one of Etzmuss et al. [16].

In a corotational formulation the elastic forces are computed in a local, unrotated coordinate frame (see Fig. 2). Therefore, we must extract the rotational part of the deformation. In order to get the rotation matrix of a triangular element, we could determine the three-dimensional transformation $\mathbf{x} = \mathbf{A}\mathbf{x}_0$ of an undeformed point $\mathbf{x}_0$ to its corresponding deformed position $\mathbf{x}$ and then extract the rotational part of **A**. However, we are only interested in the rotational transformation in the plane of the triangle. Therefore, we first project the undeformed and deformed vertices in the two-dimensional space of the corresponding plane (see Fig. 2(a)). For a triangular element with the vertex indices a, b, c and the normal $\mathbf{n} = (\mathbf{x}^b - \mathbf{x}^a) \times (\mathbf{x}^c - \mathbf{x}^a)$ we determine the plane vectors (see Fig. 2(b))

$$\mathbf{p}_x = \frac{\mathbf{x}^b - \mathbf{x}^a}{\|\mathbf{x}^b - \mathbf{x}^a\|}, \quad \mathbf{p}_y = \frac{\mathbf{n} \times \mathbf{p}_x}{\|\mathbf{n} \times \mathbf{p}_x\|} \tag{11}$$

to define the projection matrix

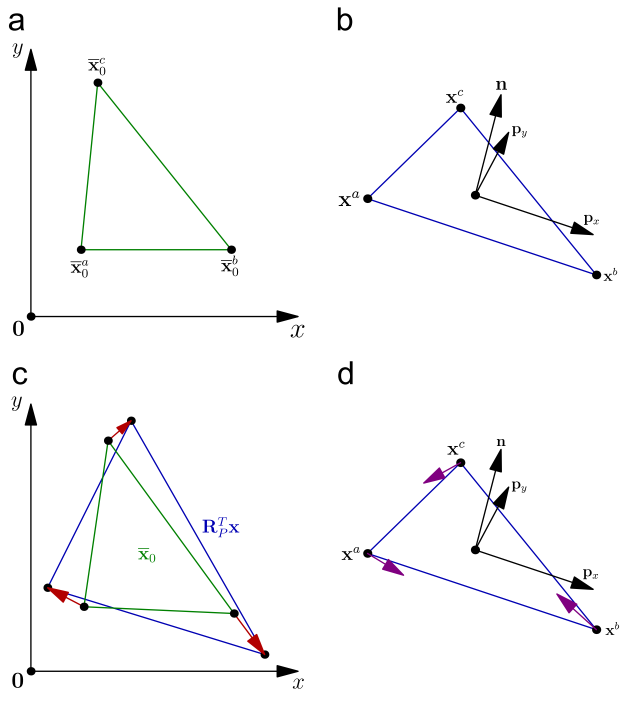
$$\mathbf{P} = \begin{pmatrix} \mathbf{p}_x^T \\ \mathbf{p}_y^T \end{pmatrix} \in \mathbb{R}^{2\times 3}. \tag{12}$$

The projection matrix $\mathbf{P}_0$ for the undeformed triangle is defined analogously. Two-dimensional coordinates can now be computed by $\overline{\mathbf{x}} = \mathbf{P}\mathbf{x}$. By defining the matrices

$$\mathbf{S} = (\overline{\mathbf{x}}_0^b - \overline{\mathbf{x}}_0^a, \overline{\mathbf{x}}_0^c - \overline{\mathbf{x}}_0^a) \tag{13}$$

$$\mathbf{T} = (\overline{\mathbf{x}}^b - \overline{\mathbf{x}}^a, \overline{\mathbf{x}}^c - \overline{\mathbf{x}}^a), \tag{14}$$

where $\mathbf{S}, \mathbf{T} \in \mathbb{R}^{2\times 2}$, we determine the matrix $\mathbf{TS}^{-1}$ which contains the transformation of an undeformed point $\overline{\mathbf{x}}_0$ to its corresponding deformed position $\overline{\mathbf{x}}$ but without the translational part. Hence, the required rotation matrix $\mathbf{R} \in \mathbb{R}^{2\times 2}$ can be extracted by a simple 2D polar decomposition [29].



a
b
c
d
y
x
0
$\overline{\mathbf{x}}_0^a$
$\overline{\mathbf{x}}_0^b$
$\overline{\mathbf{x}}_0^c$
$\mathbf{x}^a$
$\mathbf{x}^b$
$\mathbf{x}^c$
$\mathbf{n}$
$\mathbf{p}_x$
$\mathbf{p}_y$
$\mathbf{R}_P^T\mathbf{x}$
$\overline{\mathbf{x}}_0$

Fig. 2. (a) The undeformed vertices $\mathbf{x}_0$ of each element are projected in the two-dimensional space of their corresponding plane in a preprocessing step. To determine the forces acting on the vertices $\mathbf{x}$ of a triangular element, we first multiply them with $\mathbf{R}_P^T$ (see Eq. (15)). This projects the deformed vertices in the plane spanned by the vectors $\mathbf{p}_x$ and $\mathbf{p}_y$ (b) and rotates them back to an unrotated frame. (c) The difference between the resulting positions $\mathbf{R}_P^T\mathbf{x}$ and $\overline{\mathbf{x}}_0$ gives us the displacements in 2D. The forces are determined by multiplying the stiffness matrix $\mathbf{K}_e$ with the displacements. (d) Finally, we rotate the forces back to the deformed frame and transform them to the three-dimensional space using the matrix $\mathbf{R}_P$. (a) Rest state $\overline{\mathbf{x}}_0$ in 2D, (b) deformed state $\mathbf{x}$ in 3D, (c) displacements in 2D: $\mathbf{R}_P^T\mathbf{x} - \overline{\mathbf{x}}_0$, forces: $\mathbf{f}_e = \mathbf{R}_P\mathbf{K}_e(\mathbf{R}_P^T\mathbf{x} - \overline{\mathbf{x}}_0)$.

Instead of computing the forces for an element directly by Eq. (10), in the corotational formulation we first rotate a vertex back to a local coordinate frame (see Fig. 2(c)). Then the linear forces are computed and the results are transformed to the world coordinate frame (see Fig. 2(d)). This transformation can be combined with the projection matrix **P** in order to project the 3D coordinates of a vertex in the 2D space of the corresponding triangle. The transposed projection matrix $\mathbf{P}^T$ is used to transform the 2D forces of Eq. (10) to 3D. The matrix

$$\mathbf{R}_{P,e} = \begin{pmatrix} \mathbf{P}^T\mathbf{R} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^T\mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}^T\mathbf{R} \end{pmatrix} \in \mathbb{R}^{9\times 6} \tag{15}$$

combines the projection and rotation for all three vertices of a triangular element.

The corotated stiffness matrices are defined as follows:

$$\mathbf{K}_e^R = \mathbf{R}_{P,e}\mathbf{K}_e\mathbf{R}_{P,e}^T, \quad \tilde{\mathbf{K}}_e^R = \mathbf{R}_{P,e}\mathbf{K}_e, \tag{16}$$

where $\mathbf{K}_e^R \in \mathbb{R}^{9\times 9}$ and $\tilde{\mathbf{K}}_e^R \in \mathbb{R}^{9\times 6}$. Using this definition the forces for the three vertices are determined by

$$\mathbf{f}_e = \mathbf{K}_e^R\mathbf{x} + \mathbf{f}_{0,e} \quad \text{with} \quad \mathbf{f}_{0,e} = -\tilde{\mathbf{K}}_e^R\overline{\mathbf{x}}_0, \tag{17}$$

where the vector $\overline{\mathbf{x}}_0 = (\overline{\mathbf{x}}_0^a, \overline{\mathbf{x}}_0^b, \overline{\mathbf{x}}_0^c)^T \in \mathbb{R}^6$ contains the 2D positions of the undeformed triangle. Note that in contrast to Eq. (10) the resulting forces are three-dimensional since we integrated the projection in the corotational formulation.

4.2.3. *Bending*

The realistic behavior of cloth is substantially influenced by the development of folds and wrinkles. Their occurrence is highly dependent on the bending properties of the particular fabric. In order to reproduce this behavior in the case of inextensible surfaces, we employ the isometric bending model of Bergou et al. [30]. The limitation to isometric deformations has the advantage that the bending energy becomes a quadratic function in positions. Therefore, the Hessian $\mathbf{Q}$ is just a constant matrix.

For each interior edge e_i the constant Hessian $\mathbf{Q}(e_i) \in \mathbb{R}^{4\times4}$ is computed from the triangles t_0 and t_1 incident to e_i and their vertices $\mathbf{x}_i$. According to the notation given by Fig. 3, $\mathbf{Q}(e_0)$ is obtained as

$$\mathbf{Q}(e_0) = \frac{3}{A_0 + A_1}\mathbf{k}_0^T\mathbf{k}_0, \tag{18}$$

where A_i is the area of the i-th triangle and $\mathbf{k}_0$ is the row vector:

$$\mathbf{k}_0 = (c_{03}+c_{04}, c_{01}+c_{02}, -c_{01}-c_{03}, -c_{02}-c_{04}), \tag{19}$$

where $c_{jk} = \cot\angle e_j, e_k$.

Since the Hessian is assembled by considering the contributions of each interior edge, the matrix has to be updated after each subdivision or coarsening step. This update is performed locally to save computation time. Only the contributions of interior edges with changed adjacent triangles are adapted.

4.2.4. *Viscosity*

In our work we use Rayleigh damping to simulate viscosity. Rayleigh damping consists of two parts: mass and stiffness damping. The damping matrix for an element with the mass matrix $\mathbf{M}_e$ and the stiffness matrix $\mathbf{K}_e$ is determined by

$$\mathbf{D}_e = \alpha\mathbf{M}_e + \beta\mathbf{K}_e, \tag{20}$$

where α and β are the damping coefficients for mass and stiffness damping, respectively.

4.2.5. *Time integration*

For time integration the linear implicit Euler method is used together with a modified preconditioned conjugate gradient method for solving the resulting system of linear equations [7]. This provides a stable time integration of the stiff equations even for large time steps. To perform the time integration the velocity change is determined by solving the following linear system:

$$(\mathbf{M} + h\mathbf{D} + \Delta t^2\mathbf{K})\Delta\mathbf{v} = -\Delta t(\mathbf{K}\mathbf{x} + \mathbf{f}_0 - \mathbf{f}_{\text{ext}} + \Delta t\mathbf{K}\mathbf{v} + \mathbf{D}\mathbf{v}), \tag{21}$$

where Δt is the time step size and the vectors $\mathbf{f}_{\text{ext}}$, $\mathbf{x}$ and $\mathbf{v}$ contain the external forces, positions and velocities for each vertex, respectively. Matrix $\mathbf{K}$ is a sparse block matrix which is obtained by assembling the corotated element stiffness matrices $\mathbf{K}_e^R$ (see Eq. (16)) and adding the Hessian $\mathbf{Q}$ of the bending model (see Eq. (18)). Analogously the block vector $\mathbf{f}_0$ is an assembly of the vectors $\mathbf{f}_{0,e}$. After solving the linear system for $\Delta\mathbf{v}$ we get the position change by $\Delta\mathbf{x} = \Delta t(\mathbf{v} + \Delta\mathbf{v})$.

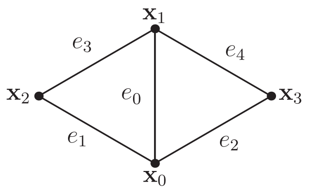


Fig. 3. Labeling of the vertices and edges which is required when determining the bending matrix $\mathbf{Q}(e_0)$ of an interior edge e_0.

4.2.6. *Efficient matrix assembly*

In general the matrix of the model is sparse. Therefore, we use a *compressed sparse row* (CSR) format to store the matrix. This format uses two arrays to describe the structure of the matrix: one array to store the column index of each element and one array to store the index offset of the first element in each row. These arrays have to be updated after each refinement or coarsening step.

The update of the matrix structure works as follows. First, we determine for each vertex the number of direct dependencies in the cloth model. Without bending, the number of dependencies of a vertex is equal to the number of outgoing edges. If we also consider our bending model, we have to add the points defined by the stencil used for computing the bending forces (see Fig. 3). Now we can construct the offset array easily. We set the first entry to 0 and the following entries are determined by a parallel prefix sum over the dependency counts of the vertices. The last value of the prefix sum gives us the number of non-zero entries in the matrix. The array for the column indices is updated in a parallel loop over all vertices. Each vertex represents a row in the matrix. We have to determine the corresponding column indices for each row. These indices are given by the dependencies of the row vertex. Finally, the indices for each row are sorted.

In order to assemble the matrix of our linear system, we first enter the masses and the mass damping terms of the particles in parallel. Then the corotated element stiffness matrices (see Eq. (16)) are determined for all faces in parallel. In the same process the stiffness damping terms are added. The terms of the bending matrix $\mathbf{Q}$ for all edges are also computed in parallel. Finally, all stiffness and bending terms are added to the linear system matrix.

5. Adaptive refinement and coarsening

The adaptive refinement of our cloth model is based on the $\sqrt{3}$–subdivision scheme of Kobbelt [2]. In order to allow for a coarsening of the mesh as well, we introduce an extension of the original scheme.

5.1. *Refinement*

The $\sqrt{3}$–refinement strategy performs a 1-to-3 split for a triangle by inserting a vertex at its center (see Fig. 4(a)–(c)). Each edge in the original mesh shared by two refined triangles is then flipped to connect the newly inserted vertices, yielding vertices with re-balanced valences (see Fig. 4(d)). If the described $\sqrt{3}$–subdivision scheme is applied two times, we get a tri-adic split where each triangle of the original mesh is subdivided into nine new triangles.

An edge on the boundary has just one adjacent triangle. Therefore, the edge flip operation is not possible for edges representing the mesh boundary. A different refinement strategy is required in this case. The goal is to get a tri-adic split after two subdivision steps not only in the interior of the mesh but also on the boundary. To reach this goal we use two different successive subdivision steps at the boundary. In the first step we perform the same subdivision as for an interior triangle but without the edge flip at the boundary edges (see Fig. 5(a)). In the second step the boundary edge is split into three equal sections by inserting two vertices and connecting them to the third vertex of the triangle (see Fig. 5(b), (c)). In the last step an edge flip is performed (see Figs. 5(d)).

This refinement strategy can be implemented in a simple recursive procedure, requiring just a generation index for each face [2]. All triangles of the base mesh have a generation of zero (see Fig. 4(a)). Inserting a vertex into a coarse triangle with even generation sets the generation of the three new triangles to the

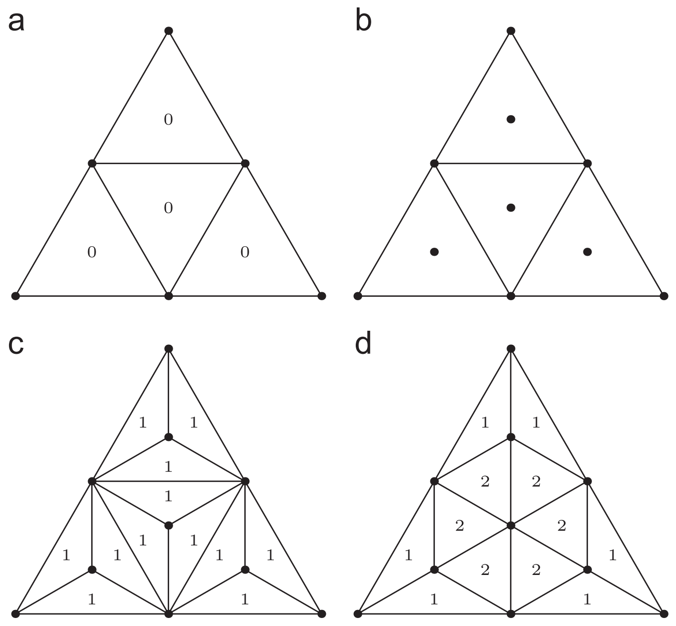


Fig. 4. $\sqrt{3}$–subdivision of a triangle mesh which is shown in (a). The numbers inside the triangles represent the generation indices of the triangles. (b) New vertices are inserted at the centers of the triangles. (c) By connecting the new vertices with the original mesh, we get new triangles. (d) Finally, an edge flip is performed for each edge where both adjacent triangles are refined. (a) Original mesh, (b) new vertices, (c) triangulation, (d) edge flip.

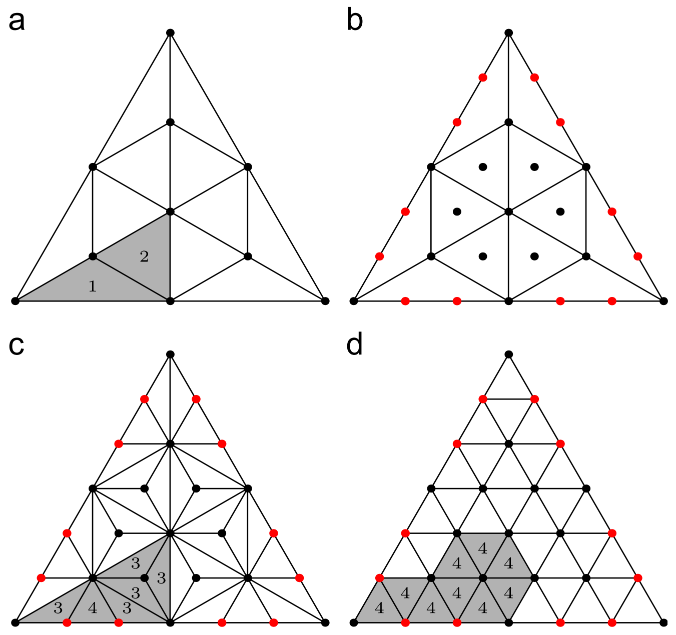


Fig. 5. The subdivision on the boundary requires two different successive steps. Only the gray triangles contain their generation index for the sake of clarity. (a) In the first step a new vertex is inserted in the center of each boundary triangle but no edge flip is performed. (b) In the second step the boundary edges are subdivided into three segments of equal length. The new boundary vertices are marked with red color. (c) Connecting the new vertices gives us a new triangulation. (d) Finally, an edge flip is performed for all vertices of generation 3. (a) First subdivision step, (b) Second step: new vertices, (c) Second step: triangulation, (d) Second step: edge flip. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

next odd integer (see Fig. 4(c)). Flipping the edge between two triangles with the same odd generation increases the generation by one (see Fig. 4(d)). Note that we restrict the difference in generation between adjacent inner triangles to one. The generation index has to be set differently when refining a boundary triangle with odd generation. Let g be the generation of the triangle to be refined. Then the generation of the new triangle sharing two of the inserted points must be set to $g+3$. The generation of the other two created triangles must be set to $g+2$. This is shown in Fig. 5. The six inner triangles have generation index $g_{inner}=2$ and the six boundary triangles are of generation $g_{boundary}=1$ after the first refinement step (see Fig. 5(a)). Now we refine the lower left boundary triangle marked in gray in Fig. 5(a). The generation of the created triangles from left to right is $g_{left}=3,\ g_{middle}=4, g_{right}=3$ (see Fig. 5(c)). The middle triangle with generation 4 has already reached its final state while the generation of the left and the right triangle is set to 3. This is necessary since we want to perform edge flips for the left and the right triangle. The left one flips the common mate edge with the adjacent boundary triangle of generation 3 and the right one the common edge with the adjacent inner triangle of generation 3 (see Fig. 5(c)). Finally, all triangles have the generation index 4 (see Fig. 5(d)).

Since the difference in generation index of neighboring triangles is restricted, the refinement of one triangle can enforce successive vertex insertions and edge flips at neighbor triangles. This keeps the valence of the vertices balanced and avoids narrow triangles. Every triangle with an odd generation has to keep track of its mate triangle which is the partner for the next edge flip. This flip is not always feasible during adaptive refinement since the neighboring triangle might be of a lower generation. In the case the refinement criterion enforces the refinement of a triangle with odd generation, first the mate triangle has to be refined and the common edge has to be flipped to fulfill the restriction on the difference in generation of neighboring triangles.

Kobbelt [2] proposes to apply a smoothing operator after each refinement step which changes the vertex positions. In our work we do not use vertex smoothing rules during the simulation since a change of vertex positions causes a change of the potential energy of the system which is not desired. This corresponds to a smoothing rule with the smoothing parameter $\alpha_n=0$ (see [2]).

5.2. Coarsening

Special care has to be taken when coarsening the triangles. Before coarsening a triangle with even generation which results in an edge flip followed by a 3-to-1 join, we have to make sure that the mate triangle is of the same generation by coarsening the mate if its generation differs (see Fig. 6(a)). Performing a 3-to-1 join requires all three participating triangles to be of the same generation. After choosing a particular triangle for coarsening we can simply identify the two neighbors as the triangles opposite of the non-mate edges (see Fig. 6(b)). The mate edge information is crucial both for flipping edges back and for performing a 3-to-1 join.

When only refining the mesh we can save the mate edge information in a face to edge table. Each time we perform a 1-to-3 split we set the edges of the coarse triangle as the face edges of the new triangles in the face to edge table to mark these edges as mate edges. But when performing a 3-to-1 join of an interior triangle it is unknown which of the three edges of the original triangle has been the mate edge. Consequently, in order to uniquely undo a 1-to-3 split, one of the three triangles resulting from such a split needs to be marked as the corresponding triangle incident to the mate edge. We store a bit for every odd generation of a triangle to mark the triangle as providing the mate edge during the 3-to-1 join. To save memory we store this bit together with the generation counter in a 32-bit integer. Using the lower 5 bits for the generation counter and the upper 27 bits for the mate edge

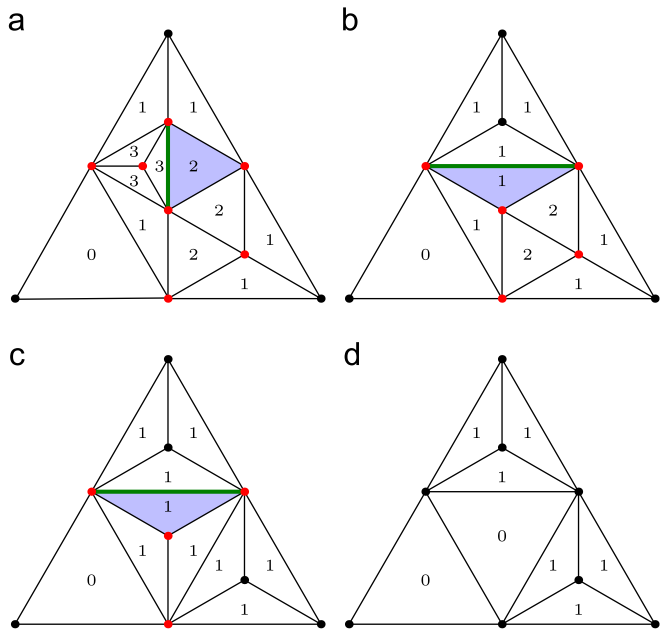


Fig. 6. Coarsening of the marked triangle in (a). Note that the mate edge of this triangle is marked with green color. The numbers represent the generation indices of the triangles. (b) First the mate triangle is coarsened and then an edge flip is performed. (c) The non-mate neighbors generation is reduced by an edge flip to the generation of the marked triangle. (d) Finally a 3-to-1 join is performed. (a) Original mesh, (b) Mate coarsening and edge flip, (c) Neighbor coarsening, (d) 3-to-1 join. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

enables us to generate 55 generations of triangles out of a base triangle by refining the same triangle successively.

5.3. *Refinement criterion*

Our refinement criterion is based on the mean curvature [27] at the mesh vertices. The mean curvature is determined by

$$\mathbf{c}(\mathbf{x}_i) = \frac{1}{2A_{\text{hybrid}}} \sum_{j \in R_1(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{x}_i - \mathbf{x}_j), \tag{22}$$

where α_{ij} and β_{ij} are the opposite angles of the edge $\mathbf{x}_i\mathbf{x}_j$ and the set $R_1(i)$ defines the one-ring of $\mathbf{x}_i$ (cf. Section 4). Note that the mean curvature for the vertices can be computed efficiently in parallel.

We take the maximum of the curvature at the three vertices of a triangle to decide whether to refine the triangle. Using the maximum allows rapid adaption if the curvature of the mesh increases locally. To steer the local refinement depth based on the mean curvature we use successively increasing limits for every refinement generation. Furthermore, we limit the maximum refinement depth by an upper bound g_{max} for the refinement generation. In our current implementation we scale the refinement limits linearly based on the difference between the limit for the maximum generation $l_{g_{max}}$ and the limit for the base generation l_{base} which are predefined, and the fraction between actual generation g and g_{max}. The refinement limit at generation g is

$$l_g = \frac{g}{g_{max}}(l_{g_{max}} - l_{base}) + l_{base}. \tag{23}$$

We use a second set of limits to decide whether to coarsen a triangle. For every refinement generation the coarsening limit is computed as a fraction of the refinement limit of the same generation. Since coarsening a triangle might also coarsen neighboring triangles (see Fig. 6) it is insufficient to purely consider the curvature at the triangles vertices. In the case of a 3-to-1 join we compare the maximum curvature of the one-ring neighborhood of the center point that is removed during the join (see Fig. 6(b)). While coarsening a triangle with even generation we build the one-ring neighborhood of the point that is removed by the 3-to-1 join after the edge flip of the triangle (see Fig. 6(a); the one-ring is marked in red).

6. Collision handling

The collision handling in our simulation is based on the idea of Bridson et al. [9]. First we perform a cloth simulation step with our continuous model (see Section 4) to advance the vertex positions from $\mathbf{x}^n$ to $\mathbf{x}^{n+1}$. Then we determine the average velocities in the vertices of the mesh by evaluating $\mathbf{v}^{n+1/2} = (\mathbf{x}^{n+1} - \mathbf{x}^n)/\Delta t$. After detecting all proximities for $\mathbf{x}^n$ the average velocities are modified by applying repulsion impulses and friction. These repulsion impulses significantly reduce the number of collisions in the following continuous collision detection step which checks the linear trajectories from $\mathbf{x}^n$ with the modified velocities $\mathbf{v}^{n+1/2}$. The resulting collisions are also resolved with friction by applying impulses. The continuous collision detection and the resolution have to be repeated until all interpenetrations are resolved in order to get a valid state of the system. To reduce the computational effort we only perform a few iterations and then use rigid impact zones, as proposed by Provot [31], to resolve all interpenetrations at once. In the end a new penetration-free state is obtained by updating the positions using the modified velocities: $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1/2}$. The final velocity is determined by solving a linear system as described in [9].

Bridson et al. use a bounding volume hierarchy (BVH) to increase the performance of the proximity and collision detection. The BVH is constructed in a precomputation step and updated in each simulation step. Since our cloth model has an adaptive resolution, a BVH would require modifications of the hierarchy in each step. Therefore, we prefer to use an acceleration method based on spatial hashing which was introduced by Teschner et al. [32].

Teschner et al. propose to use a global regular spatial grid as acceleration structure and introduce a hash function to compress this grid in a hash table. Their algorithm classifies the vertices of all objects with respect to the grid cells in a first pass. In the second pass the tetrahedrons of their volume objects are also classified. If a tetrahedron intersects a cell with vertices inside, a potential collision is reported.

In our work we use a modified variant of this method. In contrast to Teschner et al. in our approach each object has an own local spatial grid with a corresponding hash table. The size of each grid is limited by a swept bounding volume which corresponds to the object. In our work we use axis aligned bounding boxes (AABB) as swept bounding volumes which contain both the positions at the beginning and at the end of the current simulation step. The use of one grid per model has different advantages. The grid cell size influences the number of reported primitive collisions significantly. If we use large cells many primitives are mapped to the same hash value. In the case of small cells a triangle can cover many cells. In our work each model has an own cell size which yields better results for scenarios with multiple models with different resolutions. Another advantage is that the update of the grids which has to be done in each simulation step can be performed in parallel. Furthermore, we can reduce the time required for the update. A grid has only to be updated if we want to detect self collisions for the corresponding object or if its AABB intersects the AABB of another object. In the second case only one of both grids needs an update. Grids of static objects do not change, so they have not to be updated during the simulation.

The collision test starts with a bounding volume test for the simulated objects. For each collision pair which is reported by the AABB test we have to update one of the corresponding spatial grids. Each grid has a timestamp to prevent redundant updates. The following steps are performed to update a spatial grid. First the swept bounding volumes for all triangles are determined. Then we compute the indices of all cells which are intersected by the bounding volumes. If a cell is intersected, we compute a hash value for this cell and insert the corresponding triangle in the hash table. After the update a spatial grid test is performed for the primitives of the other object. This means that we determine the intersecting cells for the primitive AABBs of the other object and report the resulting collisions. The spatial grid test can be performed in parallel for all triangles since the hash table is not modified during the test.

If the spatial hashing algorithm reports a collision, we have to perform point-triangle and edge-edge tests for the corresponding triangles. Since we want to prevent redundant tests, we assign each vertex and edge to exactly one triangle. Only if this triangle is in the same grid cell as another triangle, the corresponding point-triangle and edge-edge tests are performed.

At the moment our collision detection is performed on the CPU. However, Pabst et al. [33] demonstrated that spatial hashing can be performed efficiently on a GPU. Therefore, one topic for future work will be to develop an efficient GPU implementation of the algorithm which is introduced above.

7. Results

In this section we present results with our adaptive cloth simulation method. All simulations in this section were performed on a notebook with a Intel i7-2860QM processor with four cores. In our implementation the adaption of vertex masses, the stiffness and the bending matrix as well as the computation of the mean curvature are performed in parallel. In all simulations we used a maximum of six subdivision generations. The mesh adaption is only performed in each fifth simulation step in order to reduce the additional computational costs.

In our first simulation we used the model of a curtain which is opened and closed (see Fig. 7(a)). The model has the following parameters: $E_x = E_y = E_s = 1000\ \mathrm{N/m}$, $\nu_{xy} = \nu_{yx} = 0.33$ and $\rho = 0.1\ \mathrm{kg/m^2}$. The simulation ran 20 s and the time step size was $\Delta t = 5$ ms. This model was simulated once with our adaptive mesh and once with the full resolution. A comparison is shown in the accompanying video. The visual results of both simulations are very similar. However, our adaptive model required only 5158 triangles at an average while the full resolution model had 22140 triangles. The average computation time of a simulation step with the full resolution model was 115.9 ms whereas the adaptive model required only 22.0 ms which yields a speedup factor of 5.3. Less than 5% of the computation time was required for the mesh adaption. The mean curvature of the model and the resulting triangle mesh are shown in Fig. 7(b). Fig. 8 illustrates the number of triangles of the adaptive mesh during the simulation. After 11 s the number of triangles reached its maximum. At this time the curtain was completely open and there were many wrinkles in the mesh. Then the curtain closed again and the number of triangles decreased thanks to our coarsening extension.

Another example is shown in Fig. 9. In this simulation a piece of cloth which is fixed at two vertices falls over a sphere causing several contacts with friction. This model was simulated using the same parameters as for the first model. The adaptive approach was able to reduce the number of triangles from 22 140 to 8962 at an average during the simulation. This results in a speedup factor of 2.4. For this model only 6% of the computation time was needed by the mesh adaption method.

In order to demonstrate that our method can also handle complex garment simulations, we simulated a dress on a female avatar in our last experiment (see Fig. 10(a)). In contrast to the previous experiments we used the value 5000 N/m for the Young moduli and the shear modulus which results in a very stiff model.

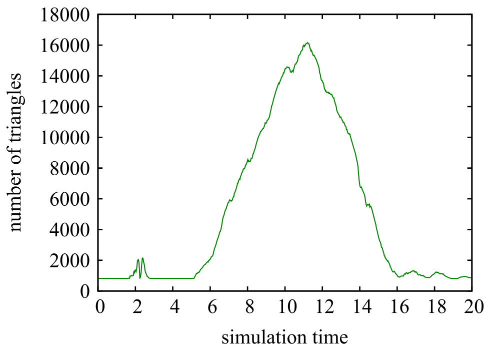


Fig. 8. The diagram shows how the number of triangles of the curtain model changes during the simulation. The number increases as the curtain starts to open after 5 s due to the resulting wrinkles. After 11 s the curtain closes again and mesh is coarsened until we reach the initial triangulation.

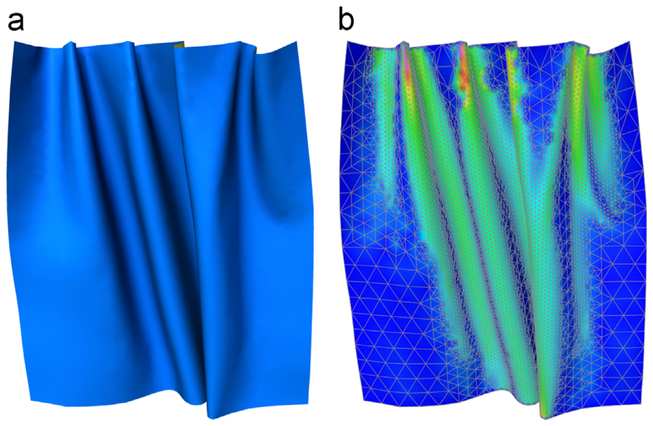


Fig. 7. Adaptive model of a curtain. (a) The surface of the opening curtain during the simulation. (b) The colors of the model represent the current mean curvature. The figure also shows the resulting triangulation. (a) Curtain model, (b) curvature and triangulation. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

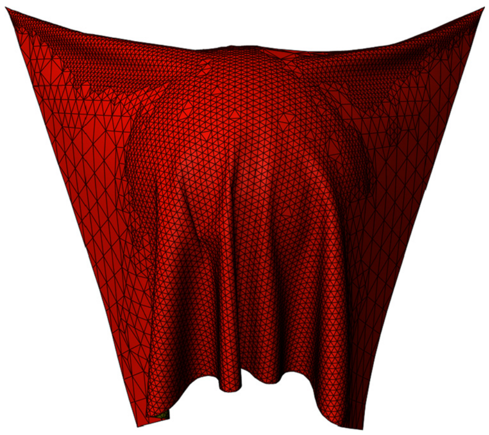


Fig. 9. Piece of cloth falling over a sphere. This example shows the adaptive remeshing of the model during a contact situation.

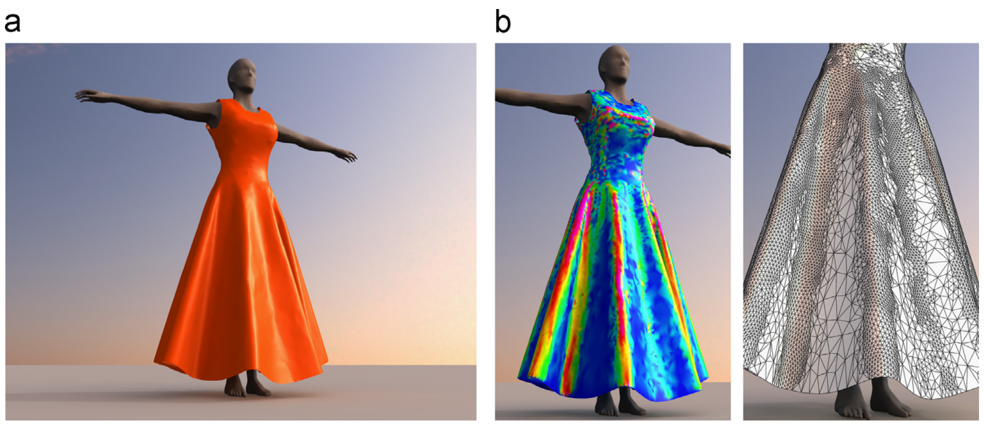


Fig. 10. Simulation of garment on an avatar model. (a) The adaptive model with folds and wrinkles during the simulation. (b) The color encoded mean curvature is shown on the surface of the model as well as the resulting adaptive triangulation. (a) Female avatar with dress, (b) Curvature and triangulation. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

The full resolution model of the dress has 66 636 triangles. In this experiment we used low refinement limits for our adaptive model since we did not want to loose much quality due to the triangle reduction. The adaptive model had 37 392 triangles at an average during the simulation. This means that the full resolution had 1.8 times more triangles than the adaptive model. The overhead of the mesh adaption was very low. Only 1.7% of the computation time in a simulation step was required for the adaption. The speedup factor of the cloth simulation without collision handling was 1.8 which exactly corresponds to the mesh reduction. However, for the collision handling we measured a speedup factor of 8.7 which is explained in the following. As described in Section 6 the continuous collision detection and the resolution have to be repeated until all interpenetrations are resolved. We used a maximum of five iterations before we resolved the remaining collisions by using rigid impact zones. In the simulation with the full resolution model we required more iterations of collision handling and the rigid impact zone method had to be applied more often. This results in the large speedup factor we measured in our experiment. The overall speedup factor of a simulation step with the dress model was 3.13 at an average. Since many simulation systems use a collision handling approach which is similar to the one of Bridson et al. [9], our adaptive refinement scheme is not only interesting for simulating cloth with a finite element method, but also for reducing the computation time of the collision handling which is still the bottle neck of most simulators.

8. Discussion

Our proposed adaptive finite element method reduces the computational effort significantly by reducing the number of elements during the simulation. We use the mean curvature to define the subdivision criterion. The speedup and the resulting loss of accuracy strongly depend on the refinement and coarsening limits that are chosen by the user. The accuracy also depends on the maximum number of subdivision generations. In our simulations we used six generations at maximum but this number can also be increased if more accuracy is required.

However, our method also has a drawback. If we use a small tolerance value for the proximity detection a coarsening step or an edge flip may result in an interpenetration. We can simply solve this problem by either increasing the tolerance value or preventing a remeshing of all triangles that are in contact. However, in future we want to provide a better solution for this problem by using a continuous collision detection based on "pseudo-trajectories" as proposed by [34].

9. Conclusion

In this article we introduced a novel adaptive cloth simulation method. The simulation mesh is refined by a $\sqrt{3}$–subdivision scheme. We extended the original scheme in order to be able to coarsen the mesh in areas where less detail is required. Our simulation model is based on continuum mechanics and we use a corotational linear FEM with triangular elements to solve the equations of motion. In contrast to a mass-spring system such a model has the advantage that the simulation converges to the correct solution when the mesh is refined. We perform the time integration with the implicit Euler method to get a stable simulation. Hence, we have to solve a linear system in each step. The problem with an adaptive model is that the matrix structure changes after each refinement or coarsening step. Therefore, we developed an efficient method to update the sparse matrix structure on the fly. In conclusion, using our adaptive model results in a significant speedup of the simulation. Furthermore, the adaption causes only a small overhead in computation time. We demonstrated this in different experiments including a complex garment simulation.

Appendix A. Shape functions

In this work we use barycentric coordinates to define the linear shape functions for our triangles. The barycentric coordinates of a point $\mathbf{m} = (x, y)^T$ in a two-dimensional triangle are defined by three linear polynomials. The first one has the following form:

$$N_1(\mathbf{m}) = \frac{1}{2A_e} \det \begin{pmatrix} 1 & 1 & 1 \\ x & x_2 & x_3 \\ y & y_2 & y_3 \end{pmatrix} \qquad \text{(A.1)}$$

$$N_1(\mathbf{m}) = \frac{(x_2y_3 - y_2x_3) + x(y_2 - y_3) + y(x_3 - x_2)}{2A_e}, \qquad \text{(A.2)}$$

where A_e is the area of the triangle. The other two are determined analogously. These polynomials are used as linear shape functions. Therefore, the derivatives of these functions are computed by

$$\frac{\partial \mathbf{N}_e}{\partial \mathbf{m}} = \frac{1}{2A_e} \begin{pmatrix} y_2 - y_3 & x_3 - x_2 \\ y_3 - y_1 & x_1 - x_3 \\ y_1 - y_2 & x_2 - x_1 \end{pmatrix}, \qquad \text{(A.3)}$$

where $\mathbf{N}_e = (N_1, N_2, N_3)^T$.

Appendix B. Supplementary material

Supplementary data associated with this article can be found in the online version of http://dx.doi.org/10.1016/j.cag.2013.04.008.

References

[1] Nealen A, Mueller M, Keiser R, Boxerman E, Carlson M. Physically based deformable models in computer graphics. Comput Graph Forum 2006;25 (4):809–36.
[2] Kobbelt L. v3-subdivision. In: SIGGRAPH '00: proceedings of the 27th annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM Press; 2000. p. 103–12.
[3] Bender J, Deul C. Efficient cloth simulation using an adaptive finite element method. In: Virtual reality interactions and physical simulations (VRIPhys). Darmstadt (Germany): Eurographics Association; 2012. p. 21–30.
[4] Magnenat-Thalmann N, Volino P. From early draping to haute couture models: 20 years of research. Vis Comput 2005;21:506–19.
[5] Choi KJ, Ko HS. Research problems in clothing simulation. Comput Aided Des 2005;37(6):585–92.
[6] Provot X. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In: Davis WA, Prusinkiewicz P, editors. Graphics interface. Canadian Human-Computer Communications Society; 1995. p. 147–54.
[7] Baraff D, Witkin A. Large steps in cloth simulation. In: SIGGRAPH '98: proceedings of the 25th annual conference on computer graphics and interactive techniques. 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, Fedkiw R, Anderson J. Robust treatment of collisions, contact and friction for cloth animation. In: SIGGRAPH '02: proceedings of the 29th annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM; 2002. p. 594–03.
[10] Goldenthal R, Harmon D, Fattal R, Bercovier M, Grinspun E. Efficient simulation of inextensible cloth. ACM Trans Graph 2007;26(3).
[11] English E, Bridson R. Animating developable surfaces using nonconforming elements. ACM Trans Graph 2008;27.
[12] Bender J, Diziol R, Bayer D. Simulating inextensible cloth using locking-free triangle meshes. In: Virtual reality interactions and physical simulations (VRIPhys). Lyon (France): Eurographics Association; 2011. p. 11–7.
[13] Bender J, Bayer D. Parallel simulation of inextensible cloth. In: Virtual reality interactions and physical simulations (VRIPhys). Grenoble (France): Eurographics Association; 2008. p. 47–6.
[14] Thomaszewski B, Pabst S, Straßer W. Continuum-based strain limiting. Comput Graph Forum 2009;28(2):569–76.
[15] Etzmuss O, Gross J, Strasser W. Deriving a particle system from continuum mechanics for the animation of deformable objects. IEEE Trans Vis Comput Graph 2003;9(4):538–50.
[16] 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; 2003. p. 244.
[17] Thomaszewski B, Wacker M, Strasser W. A consistent bending model for cloth simulation with corotational subdivision finite elements. In: Proceedings of the 2006 ACM SIGGRAPH/eurographics symposium on computer animation. SCA '06. Aire-la-Ville, Switzerland: Eurographics Association; 2006. p. 107–16.
[18] Volino P, Magnenat-Thalmann N, Faure F. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. ACM Trans Graph 2009;28 (4):105:1–16.
[19] Hutchinson D, Preston M, Hewitt T. Adaptive refinement for mass/spring simulations. In: Proceedings of the Eurographics workshop on computer animation and simulation '96. New York, NY, USA: Springer-Verlag New York, Inc.; 1996. p. 31–45.
[20] Villard J, Borouchaki H. Adaptive meshing for cloth animation. Eng Comput 2005;20(4):333–41.
[21] Li L, Volkov V. Cloth animation with adaptively refined meshes. In: ACSC '05: proceedings of the 28th Australasian conference on computer science. Darlinghurst, Australia: Australian Computer Society, Inc.; 2005. p. 107–13.
[22] Lee Y, Yoon SE, Oh S, Kim D, Choi S. Multi-resolution cloth simulation. Comput Graph Forum (Pacif Graph) 2010;29(7).
[23] Brochu T, Edwards E, Bridson R. Efficient geometrically exact continuous collision detection. ACM Trans Graph 2012;31(4).
[24] Grinspun E, Krysl P, Schröder P. Charms: a simple framework for adaptive simulation. In: SIGGRAPH '02: Proceedings of the 29th annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM; 2002. p. 281–90.
[25] Debunne G, Desbrun M, Barr AH, Cani MP. Interactive multiresolution animation of deformable models. In: Eurographics workshop on computer animation and simulation (EGCAS), 1999.
[26] Debunne G, Desbrun M, Cani MP, Barr AH. Dynamic real-time deformations using space & time adaptive sampling. In: Proceedings of the 28th annual conference on computer graphics and interactive techniques. SIGGRAPH '01. New York, NY, USA: ACM; 2001. p. 31–6.
[27] Meyer M, Desbrun M, Schröder P, Barr AH. Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege HC, Polthier K, editors. Visualization and mathematics, vol. III. Heidelberg: Springer-Verlag; 2003. p. 35–57.
[28] Müller M, Gross M. Interactive virtual materials. In: Proceedings of graphics interface 2004, GI '04, School of Computer Science, University of Waterloo. Waterloo, ON, Canada: Canadian Human-Computer Communications Society; 2004. p. 239–46.
[29] Shoemake K, Duff T. Matrix animation and polar decomposition. In: Proceedings of the conference on Graphics interface '92. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1992. p. 258–64.
[30] Bergou M, Wardetzky M, Harmon D, Zorin D, Grinspun E. A quadratic bending model for inextensible surfaces. In: Proceedings of the fourth Eurographics symposium on Geometry processing. SGP '06. Aire-la-Ville, Switzerland: Eurographics Association; 2006. p. 227–30.
[31] Provot X. Collision and self-collision handling in cloth model dedicated to design garment. Graph Interface 1997:177–89.
[32] Teschner M, Heidelberger B, Müller M, Pomerantes D, Gross MH. Optimized spatial hashing for collision detection of deformable objects. In: Ertl, T. editor. Proceedings of the vision, modeling, and visualization conference (VMV). Aka GmbH; 2003. p. 47–54.
[33] Pabst S, Koch A, Strasser W. Fast and scalable CPU/GPU collision detection for rigid and deformable surfaces. Comput Graph Forum 2010;29(5).
[34] Brochu T, Bridson R. Robust topological operations for dynamic explicit surfaces. SIAM J Sci Comput 2009;31(4):2472–93.