Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

Efficient cloth simulation using an adaptive finite element method

Jan Bender and Crispin Deul

Graduate School CE, TU Darmstadt, Germany

Abstract

In this paper we present an efficient adaptive cloth simulation based on the $\sqrt{3}$-refinement scheme. Our adaptive cloth model can handle arbitrary triangle meshes and is not restricted to regular grid meshes which are required by other methods. Previous works on adaptive cloth simulation often use discrete cloth models like mass-spring systems in combination with a specific subdivision scheme. The problem of such models is that the simulation does not converge to the correct solution as the mesh is refined. We propose to use a cloth model which is based on continuum mechanics since continuous models do not have this problem. In order to perform an efficient simulation we use a linear elasticity model in combination with a corotational formulation.

The $\sqrt{3}$-subdivision scheme has the advantage 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 only defines a mesh refinement. Therefore, we introduce an extension to support the coarsening of our simulation model as well. Our proposed mesh adaption can be performed efficiently and therefore does not cause much overhead. In this paper we will show that a significant performance gain can be achieved by our adaptive method.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation

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 the 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 paper 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 [NMK*06]. 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 [Kob00]. This scheme defines how a triangle mesh can be refined adaptively while maintaining a high mesh quality. In this paper 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 per-

DOI: 10.2312/PE/vriphys/vriphys12/021-030

formed very efficiently. Hence, the computational overhead caused by the adaption is low.

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 [MTV05, CK05]). 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 [Pro95] 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 [BW98] 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. [Pro95, CK02]. A semi-implicit method was used by Choi and Ko [CK02] 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. [BFA02] applied corrective impulses to the velocities of the particles. Goldenthal et al. [GHF*07] presented an approach based on Lagrangian mechanics in combination with a fast projection method in order to simulate inextensible cloth. English and Bridson [EB08] 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. [BDB11] combined this technique with an impulse-based approach to simulate models with hard constraints more efficiently. A continuum-based strain limiting method was introduced by Thomaszewski et al. [TPS09].

In the last years different authors proposed to use a continuous model 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. [EGS03] 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 [EKS03]. Thomaszewski et al. [TWS06] 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. [VMTF09] 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 such simulations by using an adaptive refinement of the simulation model. Hutchinson et al. [HPH96] 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 [VB05]. Li and Volkov [LV05] presented an adaptive version of Baraff's cloth simulation method [BW98] 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. [LYO*10] 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. [BEB12] use the continuous cloth model proposed by Etzmuss et al. [EGS03] 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. [GKS02] 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 [DDBC99, DDCB01].

3. Overview

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 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 $\mathbf{M}$. 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 this particle. In our work the area corresponding to a particle A_p is bounded by the midpoints of the incident edges and the barycenters of the adjacent triangles (see figure 1).

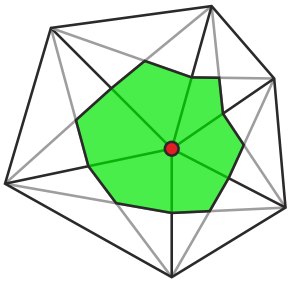


Figure 1: *The area on the surface of the triangle mesh which corresponds to a particle (red point). This area is bounded by the midpoints of the incident edges and the barycenters of the adjacent triangles.*

This area can be determined by a sum over the adjacent triangles:

$$A_p = \frac{1}{3} \sum_{\text{adj. tri.}} A_i$$

where A_i is the area of the i-th adjacent triangle. Instead of using the barycenter of a triangle one could also use the circumcenter since it is equidistant from all the vertices of the triangle. In this case the resulting area corresponds to the Voronoi region of the vertex. For our simulation we used the barycenter regions since the usage of Voronoi regions requires more computation time and a special treatment for obtuse triangles [MDSB03].

The final mass of a particle m_p is determined by the product of its corresponding area A_p with a user-defined factor ρ in order to account for the density of the simulated object

$$m_p = A_p \rho.$$

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:

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

where $\mathbf{M}$ is the mass matrix with the masses of the particles on the diagonal, $\mathbf{D}$ is a damping matrix and $\mathbf{K}$ 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 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.

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 $N_i(\mathbf{m})$:

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

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

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

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

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

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

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

where $\mathbf{B}_e = (\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3)$ and $\mathbf{C}$ 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}:

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

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:

$$\bar{\mathbf{f}}_e = A_e\mathbf{B}_e^T\mathbf{C}\mathbf{B}_e\hat{\mathbf{u}} = \mathbf{K}_e\hat{\mathbf{u}} \tag{1}$$

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 $\bar{\mathbf{f}}_e \in \mathbb{R}^6$ contains the forces for the three vertices in material coordinates.

Corotational formulation A simulation with a linear elasticity model can be performed efficiently and stable with an implicit integration scheme. However, such a model is not suitable for large rotational deformations since nonlinear effects can cause undesired deformations [MG04]. To solve this problem we use a corotational formulation similar to the one of Etzmuss et al. [EKS03].

In a corotational formulation the elastic forces are computed in a local unrotated coordinate frame. 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 $\mathbf{A}$. 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. 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

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

to define the projection matrix

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

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

$$\begin{aligned}\mathbf{S} &= \left(\bar{\mathbf{x}}_0^b - \bar{\mathbf{x}}_0^a, \bar{\mathbf{x}}_0^c - \bar{\mathbf{x}}_0^a\right)\\ \mathbf{T} &= \left(\bar{\mathbf{x}}^b - \bar{\mathbf{x}}^a, \bar{\mathbf{x}}^c - \bar{\mathbf{x}}^a\right)\end{aligned}$$

where $\mathbf{S}, \mathbf{T} \in \mathbb{R}^{2\times 2}$, we determine the matrix $\mathbf{T}\mathbf{S}^{-1}$ which contains the transformation of an undeformed point $\bar{\mathbf{x}}_0$ to its corresponding deformed position $\bar{\mathbf{x}}$ 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 [SD92].

Instead of computing the forces for an element directly by equation (1), in the corotational formulation we first rotate a vertex back to a local coordinate frame. Then the linear forces are computed and the results are transformed to the world coordinate frame. This transformation can be combined with the projection matrix $\mathbf{P}$ 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 equation (1) to 3D. The matrix

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

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

The corotated stiffness matrices are defined as follows:

$$\mathbf{K}_e^R = \mathbf{R}_{P,e}\mathbf{K}_e\mathbf{R}_{P,e}^T, \quad \tilde{\mathbf{K}}_e^R = \mathbf{R}_{P,e}\mathbf{K}_e \tag{2}$$

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

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

where the vector $\bar{\mathbf{x}}_0 = (\bar{\mathbf{x}}_0^a, \bar{\mathbf{x}}_0^b, \bar{\mathbf{x}}_0^c)^T \in \mathbb{R}^6$ contains the 2D positions of the undeformed triangle. Note that in contrast to equation (1) the resulting forces are three-dimensional since we integrated the projection in the corotational formulation.

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. [BWH*06]. 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.

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 a changed adjacent triangle are adapted.

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 [BW98]. 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:

$$\left(\mathbf{M} + h\mathbf{C} + \Delta t^2 \mathbf{K}\right) \Delta \mathbf{v} = -\Delta t \left(\mathbf{K}\mathbf{x} + \mathbf{f}_0 - \mathbf{f}_{\text{ext}} + \Delta t \mathbf{K}\mathbf{v} + \mathbf{C}\mathbf{v}\right)$$

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. Matrix $\mathbf{K}$ is a sparse block matrix which is obtained by assembling the corotated element stiffness matrices $\mathbf{K}_e^R$ (see equation (2)) and adding the Hessian $\mathbf{Q}$ of the bending model (see above). 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 $\Delta \mathbf{x} = \Delta t(\mathbf{v} + \Delta \mathbf{v})$.

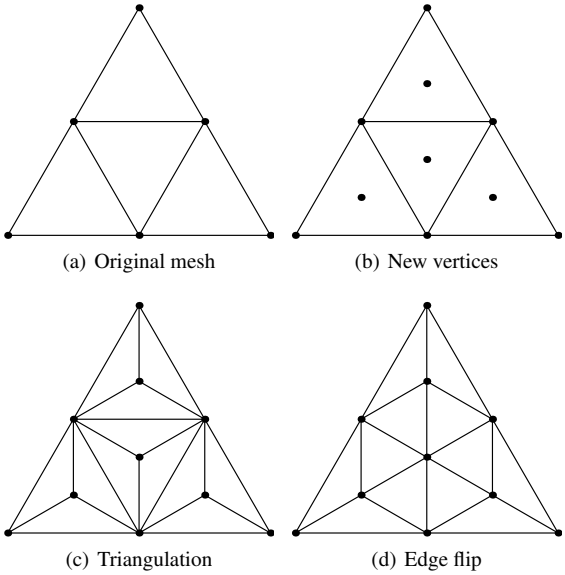
5. Adaptive refinement and coarsening

The adaptive refinement of our cloth model is based on the $\sqrt{3}$-subdivision scheme of Kobbelt [Kob00]. In order to allow for a coarsening of the mesh as well, we introduce an extension of the original scheme.

Refinement The $\sqrt{3}$-refinement strategy performs a 1-to-3 split for a triangle by inserting a vertex at its center (see figure 2). 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. 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 in nine new triangles.

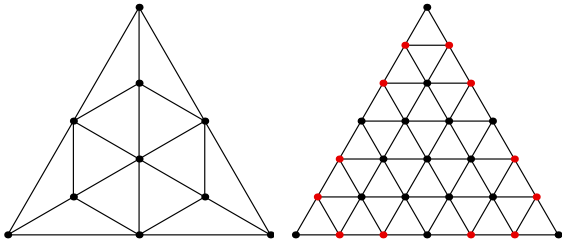
An edge on the boundary has just one adjacent triangle. Therefore, the edge flip operation is not possible for edges representing the mesh boundary. A different refinement strategy is required in this case. The goal is to get a tri-adic split after two subdivision steps not only in the interior of the 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 figure 3(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. In figure 3(b) which shows this step the inserted vertices on the boundary are marked with red color.

This refinement strategy can be implemented in a simple recursive procedure, requiring just a generation index for each face [Kob00]. All triangles of the base mesh have a gen-



(a) Original mesh (b) New vertices (c) Triangulation (d) Edge flip

Figure 2: *$\sqrt{3}$-subdivision of a triangle mesh which is shown in (a). (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) First subdivision step (b) Second subdivision step

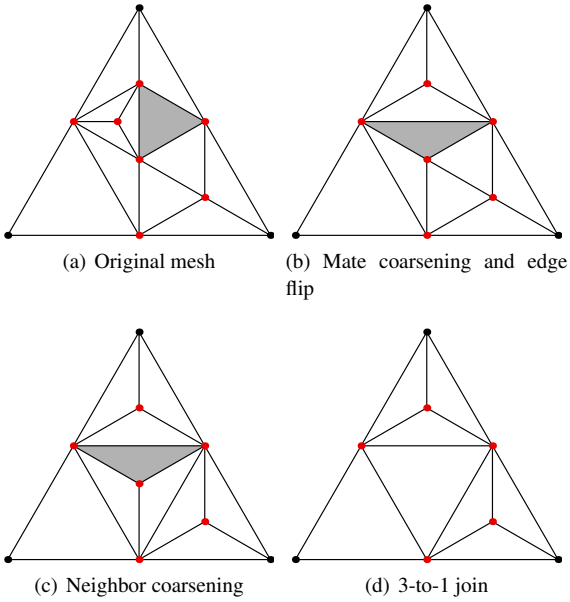
Figure 3: *The subdivision on the boundary requires two different successive steps. (a) In the first step a new vertex is inserted in the center of the boundary triangle but no edge flip is performed. (b) In the second step the boundary edges are subdivided into three segments of equal length.*

eration of zero. Inserting a vertex into a coarse triangle with even generation sets the generation of the three new triangles to the next odd integer. Flipping the edge between two triangles with the same odd generation increases the generation by one. We restrict the difference in generation between adjacent internal triangles to one. As a result 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. Since the generation is restricted, 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 [Kob00] 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 [Kob00]).

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 the mate triangle is of the same generation by coarsening the mate if its generation differs (see figure 4(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 figure 4(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 enables us to generate 55 generations of triangles out of a base triangle by refining the same triangle successively.



(a) Original mesh (b) Mate coarsening and edge flip

(c) Neighbor coarsening (d) 3-to-1 join

Figure 4: *Coarsening of the marked triangle in (a). (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.*

Refinement criterion Our refinement criterion is based on the mean curvature [MDSB03] at the mesh vertices. 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:

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

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 figure 4) 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 figure 4(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 figure 4(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. [BFA02]. 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 [Pro97], 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 [BFA02].

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. [THM*03].

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.

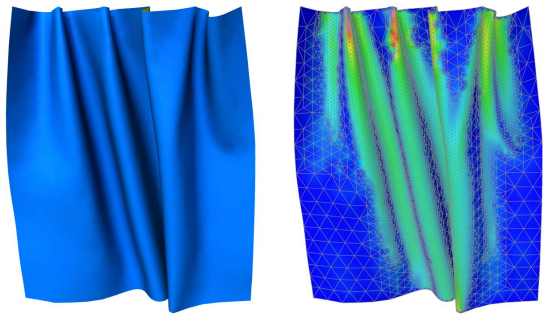
The collision test starts with a bounding volume test for the simulated objects. For each collision pair which is reported by the AABB test we have to update one of the corresponding spatial grids. Each grid has a timestamp to prevent redundant updates. The following steps are performed to update a spatial 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. [PKS10] 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.



(a) Curtain model (b) Curvature and triangulation

Figure 5: *Adaptive model of a curtain. (a) shows 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.*

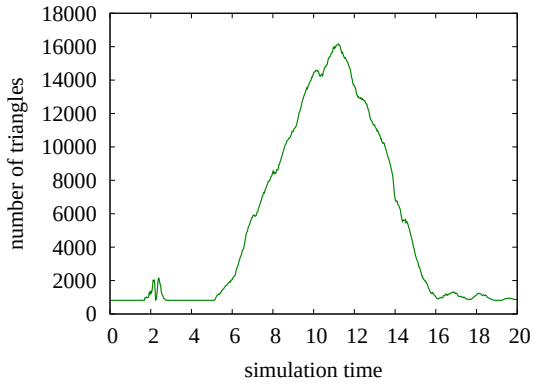


Figure 6: *The diagram shows how the number of triangles of the curtain model changes during the simulation. The number increases as the curtain starts to open after 5 seconds due to the resulting wrinkles. After 11 seconds the curtain closes again and mesh is coarsened until we reach the initial triangulation.*

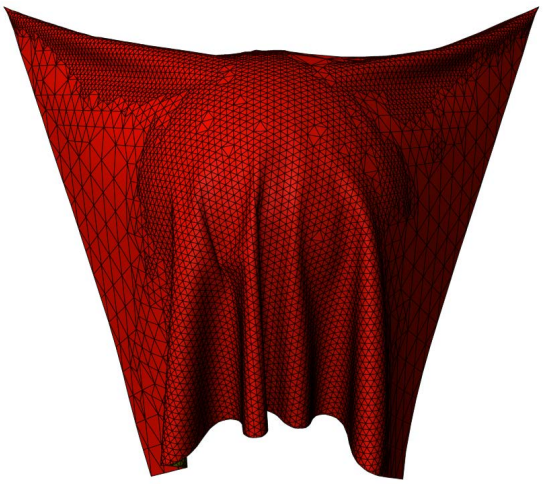


Figure 7: *Piece of cloth falling over a sphere. This example shows the adaptive remeshing of the model during a contact situation.*

In our first simulation we used the model of a curtain which is opened and closed (see figure 5(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 $\Delta t = 5\,\mathrm{ms}$. 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 percent of the computation time was required for the mesh adaption. The mean curvature of the model and the resulting triangle mesh is shown in figure 5(b). Figure 6 illustrates the number of triangles of the adaptive mesh during the simulation. After 11 seconds 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 figure 7. 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 22140 to 8962 at average during the simulation. This results in a speedup factor of 2.4. For this model only 6 percent of the computation time was needed by the mesh adaption method.

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 depends 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 preventing a remeshing of all triangles that are in contact but 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 [BB09].

8. Conclusion

In this paper 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 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. Using our adaptive model results in a significant speedup of the simulation. Furthermore, the adaption causes only a small overhead in computation time.

References

[BB09] BROCHU T., BRIDSON R.: Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput. 31*, 4 (June 2009), 2472–2493. URL: http://dx.doi.org/10.1137/080737617, doi:10.1137/080737617. 9

[BDB11] BENDER J., DIZIOL R., BAYER D.: Simulating inextensible cloth using locking-free triangle meshes. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Lyon (France), 2011), pp. 11–17. 2

[BEB12] BROCHU T., EDWARDS E., BRIDSON R.: Efficient geometrically exact continuous collision detection. *ACM Trans. Graph. 31*, 4 (July 2012), 96:1–96:7. URL: http://doi.acm.org/http://doi.acm.org/10.1145/2185520.2185592, doi:http://doi.acm.org/10.1145/2185520.2185592. 2

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 594–603. doi:http://doi.acm.org/10.1145/566570.566623. 2, 7

[BW98] 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, 1998), ACM, pp. 43–54. doi:http://doi.acm.org/10.1145/280814.280821. 2, 5

[BWH*06] 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* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 227–230. URL: http://dl.acm.org/citation.cfm?id=1281957.1281987. 5

[CK02] CHOI K.-J., KO H.-S.: Stable but responsive cloth. *ACM Transactions on Graphics 21*, 3 (2002), 604–611. doi:http://doi.acm.org/10.1145/566654.566624. 2

[CK05] CHOI K.-J., KO H.-S.: Research problems in clothing simulation. *Computer Aided Design 37*, 6 (2005), 585–592. doi:http://dx.doi.org/10.1016/j.cad.2004.11.002. 2

[DDBC99] DEBUNNE G., DESBRUN M., BARR A. H., CANI M.-P.: Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (Sep 1999). URL: http://www-evasion.imag.fr/Publications/1999/DDBC99. 2

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 31–36. URL: http://doi.acm.org/10.1145/383259.383262, doi:10.1145/383259.383262. 2

[EB08] ENGLISH E., BRIDSON R.: Animating developable surfaces using nonconforming elements. *ACM Trans. Graph. 27* (August 2008), 66:1–66:5. URL: http://doi.acm.org/10.1145/1360612.1360665, doi:http://doi.acm.org/10.1145/1360612.1360665. 2

[EGS03] ETZMUSS O., GROSS J., STRASSER W.: Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Transactions on Visualization and Computer Graphics 9*, 4 (Oct. 2003), 538–550. URL: http://dx.doi.org/10.1109/TVCG.2003.1260747, doi:10.1109/TVCG.2003.1260747. 2

[EKS03] 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* (Washington, DC, USA, 2003), PG '03, IEEE Computer Society, pp. 244–. URL: http://dl.acm.org/citation.cfm?id=946250.946946. 2, 4

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Transactions on Graphics 26*, 3 (2007), 49. doi:http://doi.acm.org/10.1145/1276377.1276438. 2

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: a simple framework for adaptive simulation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 281–290. doi:http://doi.acm.org/10.1145/566570.566578. 2

[HPH96] 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, 1996), Springer-Verlag New York, Inc., pp. 31–45. 2

[Kob00] KOBBELT L.: v3-subdivision. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, New York, USA, 2000), ACM Press, pp. 103–112. URL: http://portal.acm.org/citation.cfm?doid=344779.344835, doi:10.1145/344779.344835. 1, 5, 6

[LV05] LI L., VOLKOV V.: Cloth animation with adaptively refined meshes. In *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science* (Darlinghurst, Australia, Australia, 2005), Australian Computer Society, Inc., pp. 107–113. 2

[LYO*10] LEE Y., YOON S.-E., OH S., KIM D., CHOI S.: Multi-resolution cloth simulation. *Computer Graphics Forum (Pacific Graphics) 29*, 7 (2010). 2

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, Hege H.-C., Polthier K., (Eds.). Springer-Verlag, Heidelberg, 2003, pp. 35–57. 3, 6

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Proceedings of Graphics Interface 2004* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004), GI '04, Canadian Human-Computer Communications Society, pp. 239–246. URL: http://dl.acm.org/citation.cfm?id=1006058.1006087. 4

[MTV05] MAGNENAT-THALMANN N., VOLINO P.: From early draping to haute couture models: 20 years of research. *The Visual Computer 21* (2005), 506–519. doi:10.1007/s00371-005-0347-6. 2

[NMK*06] NEALEN A., MUELLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (December 2006), 809–836. URL: http://dx.doi.org/10.1111/j.1467-8659.2006.01000.x, doi:10.1111/j.1467-8659.2006.01000.x. 1

[PKS10] PABST S., KOCH A., STRASSER W.: Fast and Scalable CPU/GPU Collision Detection for Rigid and Deformable Surfaces. *Computer Graphics Forum 29*, 5 (2010), 1605–1612. URL: http://diglib.eg.org/EG/CGF/volume29/issue5/v29i5pp1605-1612.pdf. 7

[Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *In Graphics Interface* (1995), Davis W. A., Prusinkiewicz P., (Eds.), Canadian Human-Computer Communications Society, pp. 147–154. 2

[Pro97] PROVOT X.: Collision and self-collision handling in cloth model dedicated to design garment. *Graphics Interface* (1997), 177–189. 7

[SD92] SHOEMAKE K., DUFF T.: Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92* (San Francisco, CA, USA, 1992), Morgan Kaufmann Publishers Inc., pp. 258–264. URL: http://dl.acm.org/citation.cfm?id=155294.155324. 4

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANTES D., GROSS M. H.: Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of the Vision, Modeling, and Visualization Conference (VMV)* (2003), Ertl T., (Ed.), Aka GmbH, pp. 47–54. 7

[TPS09] THOMASZEWSKI B., PABST S., STRASSER W.: Continuum-based strain limiting. *Comput. Graph. Forum 28*, 2 (2009), 569–576. 2

[TWS06] 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* (Aire-la-Ville, Switzerland, Switzerland, 2006), SCA '06, Eurographics Association, pp. 107–116. URL: http://dl.acm.org/citation.cfm?id=1218064.1218079. 2

[VB05] VILLARD J., BOROUCHAKI H.: Adaptive meshing for cloth animation. *Engineering with Computers 20*, 4 (2005), 333–341. 2

[VMTF09] VOLINO P., MAGNENAT-THALMANN N., FAURE F.: A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph. 28*, 4 (Sept. 2009), 105:1–105:16. URL: http://doi.acm.org/10.1145/1559755.1559762, doi:10.1145/1559755.1559762. 2

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

$$N_1(\mathbf{m}) = \frac{1}{2A_e}\det\begin{pmatrix}1 & 1 & 1\\ x & x_2 & x_3\\ y & y_2 & y_3\end{pmatrix} = \frac{(x_2y_3 - y_2x_3) + x(y_2 - y_3) + y(x_3 - x_2)}{2A_e}$$

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

$$\frac{\partial \mathbf{N}_e}{\partial \mathbf{m}} = \frac{1}{2A_e}\begin{pmatrix}y_2 - y_3 & x_3 - x_2\\ y_3 - y_1 & x_1 - x_3\\ y_1 - y_2 & x_2 - x_1\end{pmatrix}$$

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