

ADAPTIVE MESHING FOR CLOTH ANIMATION

Julien Villard¹

Houman Borouchaki²

¹IFTH, Troyes, Aube, FRANCE Julien.Villard@utt.fr

²UTT-LASMIS, Troyes, Aube, FRANCE Houman.Borouchaki@utt.fr

ABSTRACT

Most of numerical simulation methods regarding cloth draping are based on mechanical models. Graphically, the representation of this model is likely to be a uniform grid. Fabrics being a very flexible material, a number of wrinkles appear on its surface when submitted to free or constrained motion (collision/applied load, supports). The problem regarding the simulation run is to represent realistically the mechanical system surface and its associated motion which are strongly related to mesh discretization. We propose a new method based on adaptive meshing allowing the mechanical system to behave without any constraint related to a uniform mesh. Numerical examples are given to show the efficiency of the method.

Keywords:adaptive meshing, cloth animation, multi-grid methods

1. INTRODUCTION

Much research in computer graphics and mechanics are focused on cloth animation. Clothes and fabrics animation consist in numerically simulating the motion of cloth in a three dimensional environment, where there are objects and/or physical phenomena (gravity, wind, and so on). Cloth is a very flexible thin material without any elastic property. Therefore, several wrinkles appear on its surface. To model a realistic piece of cloth, a fine discretization of the cloth surface is then needed. But, it brings about a lot of computational time. Most of people use coarse surface discretization and in post-processing, interpolating methods [1] to obtain cloth-like surfaces.

We propose a new cloth animation method based on a classical mechanical particles system, called “mass-spring system”. Cloth surfaces are initially discretized using uniform quad meshes where edges are oriented along the warps and the wefts directions. Each mesh node corresponds to a mass of the mechanical system. Then, in order to have an accurate representation of the surface, especially wrinkles, the initial surface discretization is adaptively refined.

This paper is organised as follows. In **section 2**, some

previous works are recalled. In **Section 3**, we present our contribution. In **Section 4**, we introduce the adaptive mesh refinement technique. In **Section 5**, we develop a new mechanical model well adapted to our refinement strategy. In **Section 6**, numerical examples are given. Finally, in **section 7**, some future works have been addressed.

2. PREVIOUS WORKS

Terzopoulos *et al.* [2] were among the first to model deformable objects using physics. Cloth is considered as a deformable object with null thickness. The authors employ the theory of elasticity to animate their deformable objects. Each object has a potential energy of deformation and its surface is discretized using a uniform grid. Potential energy is zero when the object is not out of shape. The motion equation is numerically integrated using a semi-implicit method. Given results are very interesting and they have inspired several subsequent works.

Breen *et al.* [3] have proposed a cloth draping technique based on fabric mechanical properties. Cloth surface is discretized using a uniform grid where each node is a particle of the mechanical system. For each

step of the simulation, authors let the surface free-fall: only gravity and collisions are considered. Then, system energy is minimized in order to reorganize the cloth surface. Experimental data (Kawabata) are inserted in their model, to obtain more realistic surfaces. Numerical examples were compared with experimental results and their method gives good results. Computational time is about a week on a RS6000 workstation for a 50×50 system.

Eberhardt *et al.* [4] have extended the model of Breen *et al.* adding some properties specific to fabrics, like hysteresis and anisotropic behaviours. To integrate the system, the authors used the fourth order Runge-Kutta method with an adaptive time step, that is faster than the previous one. A tablecloth draping (30×30 particles) needs about twenty minutes on a R8000 Silicon Graphics.

Provot [5] has also used a system of particles. Cloth is modeled with a network of masses linked together by massless springs. Provot has presented a new method allowing to have cloth animation with high constraints such as a hanged piece of cloth. To model the non-elastic behaviour of cloth, he used the inverse dynamic. Explicit Euler's method is applied to integrate the mechanical system. This method is interesting, because it is easy to implement. Numerous recent clothes animations are based on this model.

Carignan *et al.* [6] have extended the work made by Terzopoulos. The authors were interested in dressing virtual models. Mechanical model is derived from [2]. Nevertheless, a triangular mesh is employed for each garment's piece that they sew together. The authors have implemented the first algorithm for self-collision detection.

Volino *et al.* [7] [8] have improved the work of Carignan *et al.* [6] and that of Lafleur *et al.* [9]. Cloth surface is discretized using a coarse triangular mesh. To obtain wrinkle surfaces that look like real cloth, the mesh is interpolated in post-processing. The mechanical system is numerically solved with an explicit integration, the fourth order Runge-Kutta method. Examples show Marilyn Monroe wearing a dress and also a piece of cloth in a rotating cylinder. No information is mentioned about computational time, but the most interesting work of Volino is the collision detection algorithm that is very efficient. In recent papers [10] [11], Volino used an implicit integration method, that is faster than the explicit one.

Baraff and Witkin [12] have presented a very interesting clothes animation technique. Cloth is represented by a uniform triangular mesh. They used an implicit integration method to solve the continuum formulation of the internal energy of cloth. Integration method generates, at each time step, a sparse matrix

that is solved using a modified conjugated gradient. Furthermore, the authors have developed a technique that used an adaptive time step. Results are very interesting and computational time is very fast.

All these previous methods use uniform meshes, triangular or quadrilateral. Nevertheless, some works have been done using adaptive meshes.

Hutchinson *et al.* [13] were the first to show a multigrid method dedicated to cloth animation. Their mechanical system is the Provot one [5], a mass-springs system. On the other hand, authors employed uniform quadrilateral meshes. The main idea is to generate a coarse uniform mesh at the beginning of the simulation and to refine the mesh when angle between two edges exceed a given threshold. When this phenomenon occurs, the four quad elements sharing these two edges subdivide in sixteen smallest quad elements. It is a good idea, but the mechanical system is not well-adapted to this mesh topology and then, results are not satisfactory. What is more, computational time is very slow.

Lately, Zhang *et al.* [14] have presented a method using multilevel meshes. Their mechanical model is the Provot one. Surface is discretized using uniform triangular mesh. The authors simulate fabric draping with a coarse mesh. When the mesh is nearly at equilibrium, the algorithm refines the mesh : each triangle is subdivided into four smaller triangles, even in the place where it is not needed. This process is executed several times in order to obtain a fine uniform mesh. This method can only be used for "static" draping.

3. OUR CONTRIBUTION

The main goal is to model the mechanical behaviour of cloth when it moves. We propose a new physically based model inspired by the Provot one [5]. This model is represented by a network of particles linked together by massless springs. The mechanical properties of fabrics (stretching, shearing, bending) are modeled using springs. Stretching is the displacement of the cloth along the warps and the wefts directions. Shearing is the fabric displacement along the two diagonals directions. Finally, bending represents the curvature of the fabric surface. These three phenomena are very different and they do not react in the same way : on one hand, stretching deformation of cloth is insignificant, but in the other hand, cloth surface can be easily curved.

Our basic simulator's algorithm is an iterative process of which each iteration includes two stages :

- Forces computation : stretching, shearing, bending, gravity, wind, and so on.

- Explicit integration of the dynamic system : $\sum \vec{F} = m\vec{\gamma}$

For each simulation, the number of iterations depends on the complexity of the scene.

In our first approach, we used the model described by Provot [5]. First results seemed promising, but we noticed that this model has some drawbacks. The first one is that the results depend on the surface discretization and the second one is that it does not conserve force momentum. Therefore, we have developed a new mechanical model, especially for the bending force. This new model is well adapted to our cloth surface discretization. We detail the Provot model and our new model in **section 5**.

Geometrically, a coarse uniform mesh cannot represent an accurate fabric surface, because generally, numerous wrinkles appear on it. If we want to adequately model a fabric surface, we must employ a fine mesh. But, each mesh node corresponds to a mechanical particle and it could be difficult to numerically solve a large system in reasonable computational time. In several articles cited above, authors use system with few particles. Thus we have to make a choice:

- to use a coarse mesh and then quickly do an unrealistic cloth simulation,
- or to use a fine mesh and then slowly make a realistic cloth simulation

In order to preserve the advantages of each choice, adaptive meshes must be used. It allows us to reduce the mechanical system and to accurately represent the cloth surface. Another advantage is that no *a priori* knowledge about the cloth evolution is needed. The next section describes the adaptive refinement strategy.

4. ADAPTIVE MESHING

Cloth surface is initially defined by a coarse uniform quad mesh. Mesh edges are oriented along the warps and wefts directions and nodes are intersections between warps and wefts. When the cloth falls, the corresponding mesh must be adapted to its geometrical shape. This allows us to have an accurate surface representation all along the simulation process. An adaptive scheme consists in refining the mesh locally when the curvature exceeds a given threshold. Two problems appear :

- how to refine the mesh locally such that the warps and wefts topology is preserved.
- which mechanical model can handle mesh refinement.

To conserve warps and wefts topology, there are two different techniques. The first one consists in subdividing each curved quad element into four quad elements of which the length is half the aimed element. In this way, for each subdivided element, five new nodes are created (one in the barycenter of the element and four in the middle of its edges) if the adjacent elements were not subdivided. The second technique consists in subdividing in the same way the biggest elements sharing curved nodes. For each subdivided element, five nodes are created if it is necessary. These two methods generate non conform meshes (a node can be the middle of an edge of which the element is not subdivided). To ensure the conformity of the mesh, two kind of nodes are then used: the active ones and the virtual ones. Active nodes are the nodes for which the mesh connectivity is conforming. They share four elements and participate in mechanical simulation of cloth. Virtual nodes represent all the other nodes and are only created to ensure warps and wefts topology. They share three active nodes and their mechanical contributions are transferred on their direct neighbour nodes.

As the surface curvature is well defined at a mesh node, we consider the second refinement technique even it is more complex than the first one. To illustrate, figure 1 shows three successive applications of this technique. Initial state is four elements sharing the node P . A first refinement at P consists in subdividing the four elements sharing P in sixteen elements, eight new active nodes (the gray ones on the figure) and eight new virtual nodes (the white ones) are created. Next, a second refinement is done at a new node Q created at the previous step. Four elements are subdivided into sixteen elements; eight active nodes and eight virtual nodes are then created. Finally, a third refinement is applied at P . In this case, only the three biggest elements sharing P are subdivided. Two old virtual nodes become active, five new active nodes and six virtual nodes are then created.

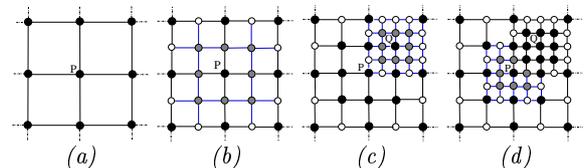


Figure 1: Mesh raffinement in three steps

4.1 Refinement criterion

The refinement criterion (at a node) is only based on geometric properties of actual cloth mesh. A refinement step is applied at a node if the local curvature of the surface exceeds a given threshold. to avoid complex computation, we approximate the local curvature

by the the deviation between the surface and the tangent plane. This deviation is computed from an estimated surface normal at the node. The normal at a node P can be approximated by the average of the eight triangle normals sharing P by considering the four direct neighbours of P and also the four barycenters of each element sharing P , as shown in figure 2. The surface deviation in P represents the biggest angle among the normal in P and the eight triangle normals described as above. For instance, this criterion is

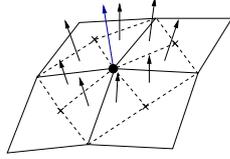


Figure 2: Computation of a node normale

only based on the geometry of the surface, but it can be useful to refine the mesh when collisions occur like in [16].

4.2 Balancing meshes

The mesh refinement criterion is purely geometric. In fact, if the angular deviation from the tangent plane at a node exceeds the given threshold, the mesh is refined at this node. The repetition of this procedure may create adjacent elements with important different sizes (as shown in figure 10) and thus directly adjacent virtual nodes. In this case, the mechanical strain transmission from virtual nodes to actives nodes becomes more complex. To avoid the creation of directly adjacent virtual nodes, the mesh can be balanced using a classical quadtree method.

5. MECHANICAL MODEL

We considere a classical model based on a network of particles linked together by massless springs. There are three kinds of springs, corresponding to three mechanical properties of cloth (stretching, shearing and bending) and we make improvements for the bending force computation. Each particle of the mechanical system corresponds to a node of the mesh and the stretching springs are materialized by the edges of the mesh. The position of each node A at time t is defined by $x_A(t)$ which is the solution of the system governed by the fundamental dynamic law:

$$\sum \vec{F}_A = m_A \vec{\gamma}_A \quad (1)$$

where m_A is the mass of the node A and $\vec{\gamma}_A$ is the acceleration of A .

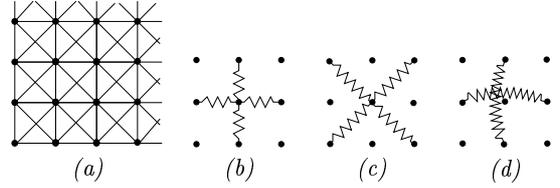


Figure 3: Mechanical model

5.1 Forces computation

The mechanical system is based on two sets of forces:

- internal forces (characteristic of cloth): stretching, shearing and bending.
- external forces (the forces generated by the three dimensional cloth environment): gravity, air resistance, wind, and so on.

5.1.1 Internal forces

To compute internal force at each node, we consider only its eight nearest neighbours which define the four quad elements sharing the node as shown in figure 3.

Stretching

Basically, Hooke's law (2) is used to compute the stretching force between a node A and one of its four neighbours B , as illustrated in figure 4.

$$F_{str}(A, B) = K_{str} \cdot (\ell - \ell^0) \quad (2)$$

where ℓ is the distance $|AB|$ at time t , ℓ^0 the distance $|AB|$ at $t = 0$ and K_{str} the stiffness of the spring linking A to B . This expression is applied in general cases

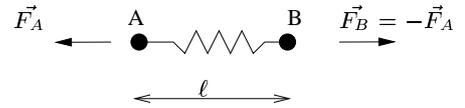


Figure 4: stretching or shearing force

and then can be used for a uniform system (springs having the same size). In the adaptive system, this equation is modified in order to take into account the variation of stiffness. When a node is added in the middle of an edge, it is equivalent to replace a spring with a size equal to ℓ , by two serial springs with size equal to $\ell/2$. Therefore, the stiffness of these two new springs must be adjusted to keep the mechanical system coherent. The following formula (3) allows us to compute stretching forces for each particle:

$$F_{str}(A, B) = K_{str} \cdot 2^{\varphi(A, B)} \cdot (\ell_{AB} - \ell_{AB}^0) \quad (3)$$

where $\varphi(AB)$ is a function that returns the level in which the spring AB belongs. The initial level corresponds to 0.

Shearing forces

Shearing forces are computed like stretching forces, only the stiffness term is different:

$$F_{she}(A, B) = K_{she} \cdot 2^{\varphi(AB)} \cdot (\ell_{AB} - \ell_{AB}^0) \quad (4)$$

where K_{she} is the shearing spring stiffness and in this case B represents a diagonally adjacent node to A .

Bending forces

In classical approach, there are two ways to represent bending forces along the warp and the weft directions. The first one is to use angular springs between the two opposite direct neighbours. The second one is to link the node by two springs : one to the second neighbour and one to the opposite second neighbour. In the first case, it seems difficult to define good reaction forces and in the second case, the second direct neighbours are taken into account which can not be handle by our refinement strategy. We propose a new approach for the bending forces computation, based on beams mechanical behaviour. As shown on the figure 5, the force modulus \vec{F} applied to the end of the beam is homogeneous to $F = \frac{EI \cdot \alpha}{\ell^2}$. If we consider that EI being equivalent to a stiffness coefficient K and that $\frac{\alpha}{\ell^2}$ equivalent to a displacement, then the expression of the force modulus is equivalent to the general equation (2). In our case, if we consider the node P and

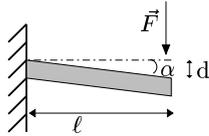


Figure 5: Bending beam

its two neighbours A and B coming into bending computation, we can say that edges PA and PB are two beams rolling the node P (cf. figure 6). When these two beams are at equilibrium, these two beams are colinear and \widehat{APB} is a π radians angle. To compute the force applied in P , we first compute the two reaction forces \vec{R}_A and \vec{R}_B respectively applied on A and B . Secondly, we easily deduce $\vec{F} = -(\vec{R}_A + \vec{R}_B)$ (cf. figure 6). Let $\vec{u}_A = \frac{\vec{PA}}{\|\vec{PA}\|}$ and $\vec{u}_B = \frac{\vec{PB}}{\|\vec{PB}\|}$ be two unit vectors and let $\vec{n} = \vec{u}_A \wedge \vec{u}_B$ be the unit normal vector to these two first vectors, then we have :

$$\vec{R}_A = K_{fie} \frac{\alpha_A + \alpha_B}{\ell_A \cdot (\ell_A + \ell_B)} (\vec{u}_A \wedge \vec{n}) \quad (5)$$

where K_{ben} is the stiffness coefficient. The reaction

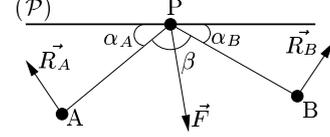


Figure 6: Our bending model

force \vec{R}_B is computed in the same manner. The interest of this computation method is that reaction forces depend on length of the edges AP and BP and the system of forces is of zero torsor. It is an important advantage of our model, because this is a multigrid model.

Another advantage is that our model is not scalability sensitive. To illustrate, two different simulations of a round tablecloth on a circular table are compared. The first simulation was done with a mesh size h and the second one with a mesh size $h/2$. The two obtained meshes are identical. Figure 7 shows these two meshes side by side and figure 8 shows these two meshes superposed.



Figure 7: Meshes with size h (top) and with size $h/2$ (bottom)

Forces applied to virtual nodes

We recall that virtual nodes are only used to ensure the mesh conformity. Therefore, the forces applied

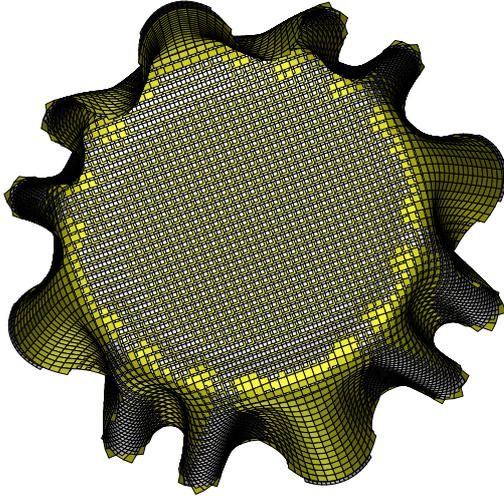


Figure 8: Two superposed meshes (underside view)

to virtual nodes are transferred to their direct active neighbours. As the mesh is balanced, the neighbours of a virtual node are active nodes. Reaction forces (stretching, shearing, bending) applied to a virtual node are uniformly distributed to neighbour nodes before the numerical integration of the system. An example is given in figure 9; it shows the repartition of the reaction forces in the case of stretching forces. The force applied on Q , \vec{F}_Q is equal to $-\vec{F}_P$ and this force is equally distributed on A and B in this manner : $\vec{F}_A = \vec{F}_B = \frac{1}{2}\vec{F}_Q$.

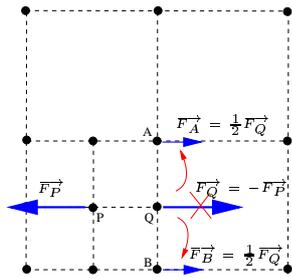


Figure 9: Translation of virtual node forces

5.1.2 External forces

To obtain realistic physical simulations, we need to add some environmental forces. Among these forces, gravity is the first one; it is applied in the same way on every active node:

$$\vec{F}_{gra}(A) = -m_A \cdot g \cdot \vec{z} \quad (6)$$

where m_A is the mass of the node A and $g = 9.81 S.I.$ is the gravity constant. We can also add a force regarding air viscosity : $\vec{F}_{air}(A) = -C \cdot \vec{v}_A$, where C is a constant.

5.2 Mass computation

Each mesh node is a particle of the mechanical system, and then each particle has a mass. Initially, before any refinement, cloth mass can be uniformly distributed on mesh nodes. When the mesh is refined, some new nodes are added and their mass must be computed. As the mesh is not uniform, the mass can not be distributed uniformly on mesh nodes. We can consider that the mass of a node is proportional to the area occupied by this node (which is computed in the same manner as for surface normal at node). The mass of an active node is equal to half the area, limited by the eight triangles sharing this node constituted by the four barycenters of each quad element and the four nearest neighbours. This computation scheme is shown in figure 10.

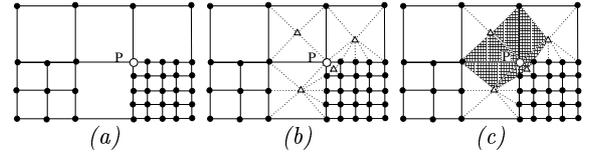


Figure 10: Mass node computation scheme

5.3 Integration method

Basically, there are two different methods to solve the fundamental equation of dynamics, implicit ones [12] [11] and explicit ones (fourth order Runge-Kutta, middle point method, and so on). In the explicit Euler's method, new positions of mesh nodes are computed in three successive stages:

$$\begin{cases} \vec{\gamma}_A(t + \delta t) &= \frac{1}{m_A} \sum \vec{F}_A(t) \\ \vec{v}_A(t + \delta t) &= \vec{v}_A(t) + \delta t \cdot \vec{\gamma}_A(t + \delta t) \\ x_A(t + \delta t) &= x_A(t) + \delta t \cdot \vec{v}_A(t + \delta t) \end{cases} \quad (7)$$

where δt is the chosen time step. A “well-adapted” time step must be chosen. if δt is too small, simulations will take a lot of computational time. If δt is too big, the system will diverge quickly. In fact, forces in the system will be important and then, velocities grow quickly and the system can diverge. To ensure the convergence of the system the velocity is controlled by bounding its modulus: if $\|\vec{v}_A\| > \xi(\nu, \delta t)$, then $\vec{v}_A = \xi(\nu, \delta t) \times \frac{\vec{v}_A}{\|\vec{v}_A\|}$, where ν is the space discretization size.

Euler's method needs a small time step to ensure a correct convergence of the mechanical step. Implicit methods seem better for this kind of simulation. Baraff *et al.* [12] have proposed an implicit method that allows to quickly animate clothes on body. Volino [11] proposed also an implicit method that seems fast. These methods need fewer iterations than the explicit one.

6. NUMERICAL EXAMPLES

In this section, we will show several examples of cloth simulation.

Collisions detection are not addressed especially for adaptive meshing with self-collision. Moreover, for all examples shown, collisions with objects (table, spheres) were resolved analytically. The simulation code is implemented in ANSI C language and the tests are realized on a J6000 Hewlett Packard workstation (bi-processors 552MHz).

The following table shows different results with different parameters like computational time, number of elements, number of levels, spatial discretization size. The number of iterations depends on the simulation.

Type	nodes (begin)	nodes (end)	levels	time
hanging	400	3843	3	4m24s
spheres	400	12437	3	30m
ball	1444	11568	4	4m15s
ball	21316	21316	0	28m

Table 1: Computational time

6.1 Hanging cloth

This example shows a hanging piece of cloth. Only two nodes of the mesh are sliding along a rigid rod with frictions. We can see on the different pictures (11 and 12) that the model is very stable even with high constraints. The initial state of the mesh is an horizontal uniform mesh; gravity does the rest of the job. The simulation is done at the 40,000th iteration.

6.2 Four spheres

In this example, we let the piece of cloth fall over four little spheres, corresponding to the four corners of a virtual square. Time needed to end this simulation is about 30 minutes, that is to obtain a free fall of the cloth. This simulation was stopped at the 30,000th iteration. Figure 13 shows some different steps of this simulation.

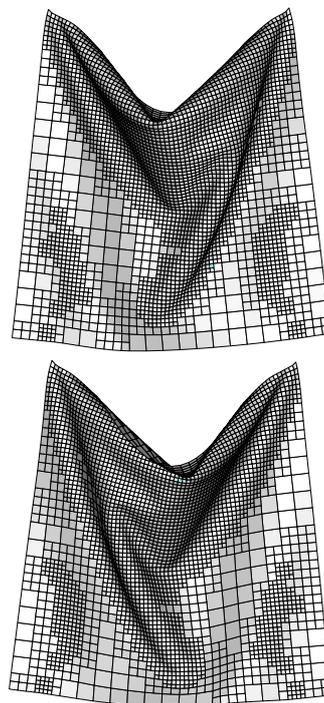


Figure 11: Hanging cloth (front and behind view)

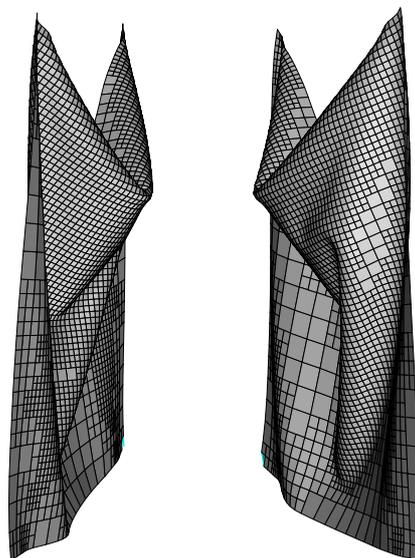


Figure 12: Hanging cloth (left and right view)

6.3 The ball

This simulation concerns the draping of a sphere. There are 4 levels of refinement that give a very smooth surface. The final result needs about 4 minutes

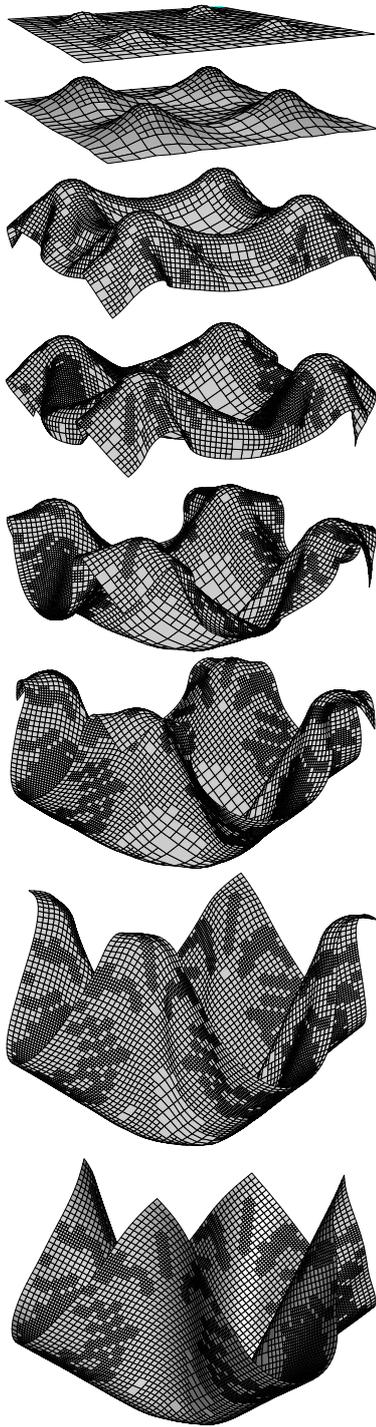


Figure 13: Cloth falling over four spheres

and 10,000 iterations. Figure 14 shows two intermediate stages and the final stage.

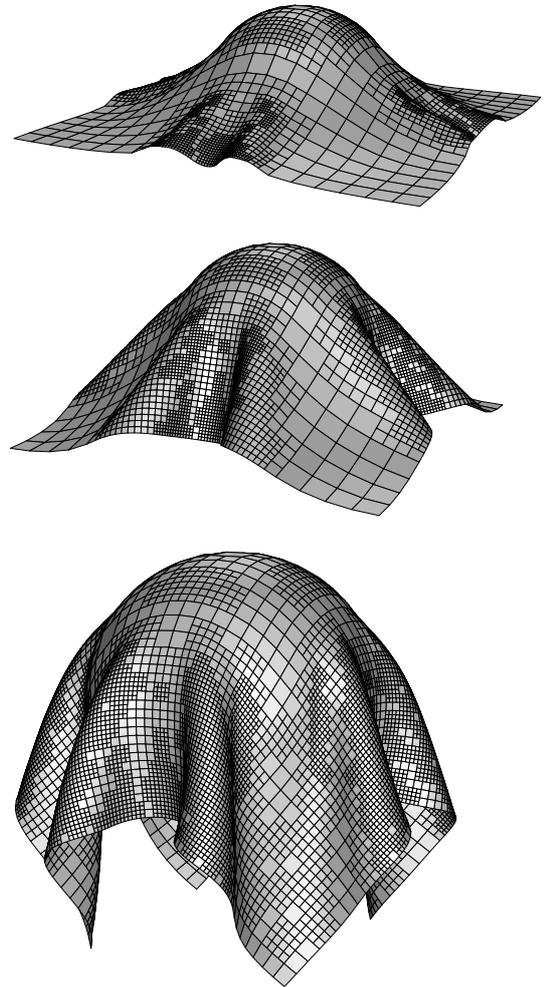


Figure 14: Square piece of cloth on a ball

6.4 Adaptive mesh versus uniform mesh

In order to prove that our method gives good results, we compare two simulations based on the previous simulation, that is, a piece of square cloth on a ball. The first one was realized with an adaptive mesh : space discretisation size of initial mesh is $20mm$ and we stick the level of refinement at 4. The second one was realized with a uniform mesh : space discretisation size is $2.5mm$. Figure 15 shows the two meshes side by side and figure 16 shows these two meshes superposed. We can see that these two meshes are nearly identical.

These two simulations were stopped after 10,000 iterations. The uniform mesh has about 21,000 nodes while adaptive mesh about 11,000 nodes. Computational time of the adaptive mesh is about 4 minutes and the uniform one needs half an hour to give the same result. Figure 17 shows the final adapted mesh rendered in a 3D scene using POV-Ray.



Figure 15: Two different simulations?



Figure 16: Two superposed meshes



Figure 17: 3D scene rendering of the ball example

7. CONCLUSIONS

In this paper, we have introduced a new method to realistically animate cloth. Using adaptive meshing allows us to reduce the number of mesh elements and

then to reduce computational time. In addition, multi-grid method are tuned with different parameters allowing to easily control each simulation; cloth surface is then more realistic than in methods employing uniform meshes.

This work is about the first stage of a cloth simulation software. Future work include mesh simplification, collision detection and implicit integration. Mesh simplification is the reverse operation of mesh refinement. This technique consists in deleting the nodes and the elements inside the area that become flat (depending on criterion) during the simulation. Collision detection is a hard work for cloth animation, because fabrics are very flexible and cloth surface could intersect itself. Our integration method is explicit and needs many iterations to numerically solve the system. We think that an implicit one can reduce computational time significantly.

References

- [1] Hadap S., Bangarter E., Volino P., Magnenat-Thalmann N. "Animating Wrinkles on Clothes." *IEEE Visualization '99*, pp. 175–182. IEEE Computer Society Press, San Francisco, USA, October 1999
- [2] Terzopoulos D., Platt J., Barr A., Fleischer K. "Elastically Deformable Models." *Computer Graphics*, vol. 21(4), pp. 205–214. July 1987
- [3] Breen D.E., House D.H., Wozny M.L. "Predicting the Drape of Woven Cloth Using Interacting Particles." *Siggraph '94*, pp. 365–372. Orlando, USA, July 1994
- [4] Eberhardt B., Weber A., Strasser W. "A Fast, Flexible, Particle-System Model for Cloth Draping." *IEEE Computer Graphics and Applications*, vol. 16, no. 5, 52–59, September 1996
- [5] Provot X. "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour." *Computer Interface Proceedings*, pp. 147–154. Quebec City, Canada, May 1995
- [6] Carignan M., Yang Y., Magnenat-Thalmann N., Thalmann D. "Dressing Animated Synthetic Actors with Complex Clothes." *Computer Graphics Proceedings*, vol. 26, pp. 99–104. ACM Siggraph, 1992
- [7] Volino P., thalmann N.M., Jianhua S., Thalmann D. "An Evolving System for Simulating Clothes on Virtual Actors." *IEEE Computer Graphics and Applications*, vol. 16, no. 5, 42–51, September 1996

- [8] Vollino P., Courchesne M., Thalmann N.M. “Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects.” *SIGGRAPH '95*, pp. 137–144. 1995
- [9] B.Lafleur, Thalmann N.M., Thalmann D. “Cloth Animation with Self-Collision Detection.” *IFIP Conference on Modeling in Computer Graphics proceedings*, pp. 179–197. Springer-Verlag, Tokyo, Japan, 1991
- [10] Pascal Volino N.M.T. “Comparing Efficiency of Integration Methods for Cloth Animation.” *Proceedings of CGI'01*. Hong-Kong, July 2001
- [11] Volino P., Magnenat-Thalmann N. “Implementing fast Cloth Simulation with Collision Response.” *Computer Graphics International 2000*, pp. 257–266. 2000
- [12] Baraff D., Witkin A. “Large Steps in Cloth Simulation.” *Computer Graphics Proceedings*, pp. 43–54. ACM Siggraph, Orlando, FL, USA, July 1998
- [13] Hutchinson D., Preston M., Hewitt T. “Adaptive refinement for mass/spring simulations.” *7th Workshop on Animation and Simulation*. Poitiers, France, 1996
- [14] Zhang D., Yuen M.M. “Cloth simulation using multilevel meshes.” *Computers and Graphics*, vol. 25, 383–389, december 2001
- [15] Ng H.N., Grimsdale R.L. “Computer Graphics Techniques for Modeling Cloth.” *IEEE Computer Graphics and Applications*, vol. 16, no. 5, 28–41, September 1996
- [16] Etmuss O., Eberhardt B., Hauth M., Strasser W. “Collision Adaptive Particle Systems.” *Proceedings Pacific Graphics 2000*, 2000