3rd Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2006)
C. Mendoza, I. Navazo (Editors)

Non structured meshes for Cloth GPU simulation using FEM

J. Rodriguez-Navarro[1] and A. Susin[1]

[1] Laboratori de Simulació Dinàmica (LABSID), Dept. Matemàtica Aplicada 1, UPC-Barcelona (Spain)

Abstract

We present a Finite Element Method (FEM) implementation for cloth simulation on the GPU. The advantages of FEM are twofold: the realism of cloth simulations using this method is improved compared with other methods like the widely used mass-spring one, and it has a wider application rank because it can be used for general triangulated cloth meshes.
We are able to detect collisions between cloth and other objects (solids or deformables) and also we deal with self cloth collisions. This is also done in the GPU using image-based collision methods. We have also improved a GPU-Gradient Conjugate method for solving the linear equation systems involved in the FEM solution.
Two more methods are also implemented in the GPU to compare with the FEM method: a mass-spring model (based on rectangular meshes) and a constraint method (based on triangular meshes).

1. Introduction

Cloth simulation can be considered as a particular case included at the more general framework of deformable models. A deformable model simulation requires the update of the model vertex positions at each time step according to some deformation law. This can be considered like a new object representation at each time step and was not possible take advantage of the old graphic cards for deformable simulations. Nowadays graphic cards incorporate programable capacity for their GPU (Graphics Processor Unity) and allows to read vertex positions from a texture in a vertex or shader program.

On the other hand, to achieve a realistic cloth simulation, one has to choose a dynamic model in order to obtain a good approximation of the actual cloth behavior. One of the most used models in the computer graphics community is the mass-spring model [Pro95]. Nevertheless this model has two main drawbacks, first it is designed for structured (or rectangular) meshes. It gives poor results when applied to general unstructured (or triangular) meshes. And second, it is difficult to simulate an actual cloth material because the elasticity parameters included in the model have no relation with the physical parameters used in the industry.

We are interested in dealing with these two drawbacks and also to implement the simulation in the GPU. This way we have decided to choose the Finite Element Method (FEM) as dynamical model using triangular elements in order to apply FEM to unstructured meshes, which gives us more flexibility when modelling a garment. Moreover, Finite Element methods are obtained directly from elasticity theory which means that material parameters are naturally included in the method.

Figure 1: *Garment simulation in the GPU for a moving humanoid.*

When the implementation is aimed to the GPU, structured meshes are easier to include in the GPU internal texture codification. If unstructured meshes are used, it can be still done but it will require an extra texture to deal with the vertex connectivity. We have implemented the FEM method in the GPU and compare its performance with the implementation, also in the GPU, of the mass-spring model and with the first GPU implementation of the Baraff and Witkin cloth constrain model [BW98] using implicit integration. This model has been considered because, like FEM, it is also defined for unstructured meshes.

The paper is organized as follows, section 2 is devoted to previous work in the area, section 3 describes the cloth simulation model and its GPU implementation. Results and conclusions are shown on last section.

2. Previous work on cloth simulation

Cloth simulation is a well known and widely studied computer graphics problem. We can roughly classify previous work according to the emphasis on modelling, simulation and computational efficiency. A good reference survey for the field is [NG96].

Early works are essentially devoted to the modelling aspects. Papers by Terzopoulos et al. [TPB87], [TF88] where the first physically-based ones. They introduce cloth simulation for graphics community as a problem of deformable surfaces and used techniques from mechanical engineering like the finite element method and energy minimisation. Other approaches on the dynamic modelling have been, the particle-based models from the works of Breen et al. [BHW94] and Eberhardt et al. [EWS96], the energy-based models from Carignan et al. [CYT92] and Baraff and Witkin [BW98]. The most successful approach in modelling has been the mass-spring one, introduced by Provot [Pro95].

One of the first, present and more successful work in cloth simulation is devoted to the MiraLab team leaded by N. Magnenat-Thalmann. Their contributions start in the early 80's and have reached many important results in all the above areas classification [Miralab].

Like in other graphics topics, the efficiency of an implementation is related both with the chosen model and the hardware performance. Moreover, a trade-off between speed and precision has to be assumed, the work of Hauth and Etzmuss [HE01], and the one of Volino and Magnenat-Thalmann [VMT01] discuss the convenience of numerical integrators. The final decision for choosing one simulation model or another is mainly related with the final application field, for video-games or films the requirements are totally different. The work of Jacobsen [Jac01] for the game industry has been the pioneer in introducing a dynamic model for the cloth together with the numerical integrator, the Verlet method, which is suitable for GPU implementations. The first GPU cloth implementation is due to S. Green [Gr03], a structured rectangular mesh cloth and a solid sphere are simulated using Verlet method on the GPU. Only stretch forces has been simulated. For dealing with collisions on the GPU image based methods are introduced by Vassilev et al. [VSC01] for walking humanoids without dealing with occlusions. Other recent papers by Kolb et al. [KJ01], [KLR04] studies collisions with complex but static objects.

Müller et al. [MDM02] presented a FEM-based approach for real-time deformations on the CPU. By estimating the rotational part of the deformation and using linear elasticity, they create plausible animations free of the disturbing artifacts present in linear models and faster than non-linear models. Teschner et al. [THM04] perform deformations on low resolution tetrahedral meshes, coupled with high resolution surface meshes used to visualize the deformed body. FEM methods applied to cloths use typically non-linear elements, because textiles are very flexible and bend easily, forcing deformation being modelled by the Green's strain tensor. Eztmuss [EKS03] introduce a FEM linear method that can be applied for large deformations using rotation correction directly deduced from the non-linear Green's tensor.

We use ideas from this last paper to implement the FEM method in the GPU and we compare it with our own GPU implementations of the other approaches: the mass-spring model [Gr03] using Verlet integrator, but taken into account also shear and bending forces; and the Baraff-Witkin [BW98] constrain model using implicit integration.

3. Model Description

In this section we will describe the method we have used for modelling cloth behavior according to elasticity laws.

The dynamics of the cloth will be determined by the Lagrange equation assuming that forces are non-conservative, [GOL50]

$$M\ddot{x} + C\dot{x} + \frac{\partial V(x)}{\partial x} = f_{ext} \tag{1}$$

where x is the position, $V(x)$ is the elastic potential, M is the mass density, C is the damping density and f_{ext} are the applied external forces.

$$\begin{aligned} \dot{x} &= v & (2) \\ M\dot{v} &= -Cv - K(x - x_0) + f_{ext}. & (3) \end{aligned}$$

Here K denotes the stiffness matrix computed from the elastic potential as it will be explained next.

The deformations produced in the cloth can be expressed in terms of the metric tensor. For in-plane deformations:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{4}$$

where u_i, u_j are displacements in the cloth and ε_{ij} denotes the deformation terms. If $i = j$ deformation represents the elongation of the cloth, whereas $i \neq j$ is related to shear deformation. Deformations out-plane are obtained using bending forces.

The FEM allows to find the solution of a differential equations system representing a model of a physical problem in a continuous medium [ZT93].

Linear elasticity uses a linear strain formulation, the Cauchy strain tensor, as well as a linear material law and yields linear partial differential equations that can be handled efficiently by an implicit time integrator. The reason why linear elasticity is not adequate to simulate cloth behavior is that a linear strain approach is only valid for small displacements or rotations. However, since textiles are very flexible and bend easily, the linear formulation, based on the Cauchy strain tensor is not valid.

3.1. The shape functions

FEM methodology is based on considering the continuous material problem as a discrete problem which is solved on a set of points (nodes) associated to the elements that decomposed the geometry domain. The solution is extended to the whole domain by interpolation using the shape polynomial functions N_i associated to each node. Now we present a different way, from the one usually found in many books, to compute the shape functions which is more suitable for GPU implementations.

In the linear interpolation case, elements are triangles and nodes are the vertex. Shape functions are linear functions defined implicitly as

$$\begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}, \tag{5}$$

where $(x_i, y_i, z_i), i = 1,2,3$ are the vertex coordinates.

One important property of the shape functions is the normalization condition, included in the first equation of (5) is written as $N_1 + N_2 + N_3 = 1$. This can be used in order to remove one of the shape functions in the above expression yielding

$$\begin{bmatrix} x - x_3 \\ y - y_3 \\ z - z_3 \end{bmatrix} = \begin{bmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \\ z_1 - z_3 & z_2 - z_3 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix},$$

that can be expressed using the notation

$e_i = (x_i - x_3, y_i - y_3, z_i - z_3)^T, \quad i = 1,2$ as

$$\begin{bmatrix} x - x_3 \\ y - y_3 \\ z - z_3 \end{bmatrix} = \begin{bmatrix} e_1 & e_2 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} \tag{6}$$

Multiplying by the transposed matrix $[e_1^t e_2^t]$ one can obtain the explicit expressions for the shape functions and their derivatives:

$$\begin{aligned} N_1 &= [(e_2^t \cdot e_2)(e_1^t \cdot X) - (e_1^t \cdot e_2)(e_2^t \cdot X)]/\det \\ N_2 &= [(e_1^t \cdot e_1)(e_2^t \cdot X) - (e_1^t \cdot e_2)(e_1^t \cdot X)]/\det \\ N_3 &= 1 - N_1 - N_2 \\ dN_1 &= [(e_2^t \cdot e_2)e_1 - (e_1^t \cdot e_2)e_2]/\det \\ dN_2 &= [(e_1^t \cdot e_1)e_2 - (e_1^t \cdot e_2)e_1]/\det \\ dN_3 &= -dN_1 - dN_2 \end{aligned}$$

where $X = (x - x_3, y - y_2, z - z_3)^t$ and $\det = (e_1^t \cdot e_1)(e_2^t \cdot e_2) - (e_1^t \cdot e_2)^2$.

As we pointed out in the previous subsection for large deformations, Cauchy's strain tensor is not valid and usually Green's stain tensor G is used. G is expressed in terms of the deformation gradient $J = (\frac{\partial D}{\partial u_1}, \frac{\partial D}{\partial u_2})$:

$$G = \frac{1}{2}(J^t J - Id).$$

The linearization of Green's tensor, with the assumption that the deformation derivatives $\frac{\partial D}{\partial u_i}$ are small, yields to Cauchy's strain tensor.

For each triangle T we compute the shape functions N_j and their derivatives $\frac{\partial N_j}{\partial u_i}$ associated to each vertex j in the rest state.

3.2. Corotational Method

As mentioned above, large rotations prevent the use of the linear strain formulation. This is due to that Cauchy's tensor is not invariant under rotations. Hence, in each time step we are going to remove these rotations before computing the deformation forces. This is known as corotational method and it is equivalent to build a rotated rest state and linearizing with respect to this rest state. For that we consider the polar decomposition of the deformation gradient , that is, the deformation gradient J can be decomposed into a rotation R and a pure deformation U.

$$J = R \cdot U \tag{7}$$

For tetrahedra elements Müller et al. [MDM02] uses a per vertex stiffness warping approach to compute the rotational part. Etmuss et al. [EKS03] introduces another CPU algorithm for triangles based on a SVD decomposition to find the involved rotation. We have implemented a triangle version for the GPU consisting of a new adaptation of the method introduced on [MDM02], [NPF05]. Instead of making a vertex based calculation, we compute a stiffness warping based on the triangle face.

If we denote by e_1 and e_2 two adjacent edges of a triangle in its rest position, we obtain an orthonormal base of the space from a Gram-Schmidt step $\tilde{e}_2 = e_2 - (e_1 \cdot e_2) \cdot e_1$ and finally using the cross product $e_3 = e_1 \times \tilde{e}_2$. This way we obtain a base $N = [e_1, \tilde{e}_2, e_3]$ of the space. Analogously, if $\tilde{N}$ denotes the corresponding base associated to the deformed triangle, then the rotational part R is computed as

$$R = N \cdot \tilde{N} \tag{8}$$

Finally we follow [EKS03] to compute the terms associated to stretch and shear forces. Here the elasticity parameters are introduced according to the type of simulated material.

3.3. Bend Forces

For simulating real cloth a bending term has to be introduced. Most bend forces in textile animation rely on some Laplacian formulation as all models based on springs lead to such a formulation [EG02]. This includes the notable work of Choi et al. [CK02]. Here, we also follow [EKS03] for compute a bending force term.

4. Integration method

For the sake of stability, to solve the dynamic system (2) we use an implicit method. We consider the system

$$\begin{aligned} \dot{x}_{i+1} &= x_i + h v_{i+1} && (9) \\ M\dot{v}_{i+1} &= M v_i + h(f_{ext} - C v_{i+1} - K(x_{i+1} - x_0)) && (10) \end{aligned}$$

Substituting (9) in (10) we get a linear system of equations

$$(M + hC + h^2 K) v_{i+1} = M v_i + h(f_{ext} - K(x_i - x_0)) \tag{11}$$

Denoting

$$A = M + hC + h^2 K$$

and

$$b = M v_i + h(f_{ext} - K(x_i - x_0))$$

(11) is reduced to the usual form $Ax = b$.

4.1. Conjugate Gradient on the GPU

The above linear system can be solved using different methods. Due to our restriction to the GPU, we have chosen an iterative method well suited for GPU implementation. Although there exists previous GPU implementations of the Conjugate Gradient method [KW03],[GPU-2], we present here an optimal implementation for present graphic cards.

The traditional gradient conjugate algorithm using orthogonal directions consists in the following steps:

$$p_0 = r_0 = b - Ax$$

for each step

$$u = A p_i$$

$$\begin{aligned} \alpha &= \frac{r_i^t \cdot r_i}{p_i^t \cdot u} \\ x_{i+1} &= x_i + \alpha p_i \\ r_{i+1} &= r_i - \alpha u && (12) \\ \text{if } (|r_{i+1}^t \cdot r_{i+1}| &< \text{ tolerance) break;} \end{aligned}$$

$$\begin{aligned} \beta &= \frac{r_{i+1}^t \cdot r_{i+1}}{r_i^t \cdot r_i} && (13) \\ p_{i+1} &= r_{i+1} - \beta p_i \end{aligned}$$

If we substitute (12) in (13), then GC has the following form

$$p_0 = r_0 = b - Ax \tag{14}$$

for each step

$$u = A p_i \tag{15}$$

$$d_r = r_i^t \cdot r_{i+1}^t,\ d_g = p_i^t \cdot u,\ d_b = r_i^t \cdot u,\ d_a = u^t \cdot u \tag{16}$$

$$\alpha = \frac{d_r}{d_g},\ \beta = \frac{d_r - 2\alpha d_b + \alpha^2 d_a}{d_r}$$

$$\begin{aligned} x_{i+1} &= x_i + \alpha p_i && (17) \\ r_{i+1} &= r_i - \alpha u && (18) \\ p_{i+1} &= r_{i+1} + \beta p_i && (19) \\ \text{if } (|\beta d_r| &< \text{ tolerance) break;} \end{aligned}$$

This new formulation of the GC method allows to a better data compactification and gives a better performance when using the GPU. We use only two framebuffer objects, in short FBO. One of them attaches the textures of unknown x, residual term r and previous unknown values p. This FBO is used to render the fragment shader which compute the equation (14) and the fragment shader which compute simultaneously equations (17), (18) and (19). We would like to point out that the output of the fragment shader is two colors (p_0 and r_0) in the first case, and three colors (x, r and p) in the second case. To do this we employ the Multi Render Target extension. Furthermore, the draw buffer and read buffer are the same texture saving in this way GPU memory and swap time between textures.

The second FBO is used to solve the equation (15) and *parallel reduction* (16), see [GPU-2] for the details.

4.2. Collision Detection

A lot of effort has been done in this field, however in most cases the collision detection is not carried out integrally on the GPU. Algorithms are divided in two groups: Collisions (cloth/object) and Self-collisions (cloth/cloth).

For the first case, we use the technique introduced in

[RSS05] inspired on [VSC01]. This method is an image based technic, consisting in grabbing the object depth-buffer in each time step such that object deformations, like a human body in our case, are also taken into account. Moreover, the collision detection is made between corresponding pieces of the object and the cloth in a fragment shader. When, based in the corresponding depth, a collision is reported, the involved cloth vertex is marked as a collision vertex for response. One of the key points for having a robust collision detection is related with the number and position of the local cameras chosen to grab the associated depth buffer object.

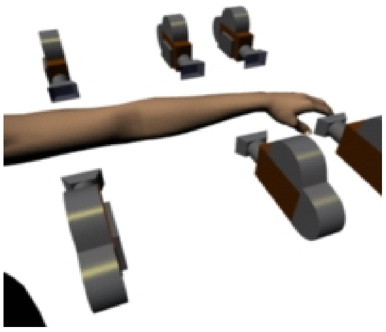


Figure 2: *Local camera position for collision detection.*

To handle self-collisions we decide to implement on GPU an optimal spatial hashing method [THM03] with the difference that we make a local voxelization for each time step only in the space covered by the cloth.

The corresponding hash function for each point $X = (x,y,z)$ is defined as

$$hash(X) = \left(\left(\frac{x-m_x}{M_x-m_x} + \frac{y-m_y}{M_y-m_y}\right)GS + \frac{z-m_z}{M_z-m_z}\right)GS.$$

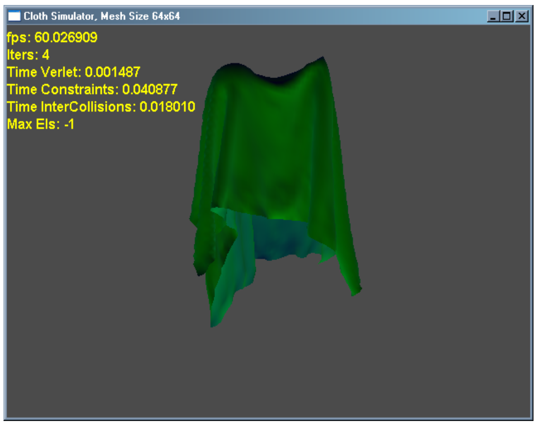
where $m = (m_x, m_y, m_z)$ and $M = (M_x, M_y, M_z)$ are the minimum and maximum respectively of the cloth in each time instant, and GS is the fixed grid size.

4.3. Collision Response

In order to achieve a stable collision response, we have adapted a solution introduced in [BW98]. This solution modify both velocity and position using an arbitrary correction term, y, introduced only to move a particle to a desired location during the backward Euler step such that small jumps are filtered. This correction term modifies expression (2) to the following one

$$\begin{aligned} \dot{x} &= v + y \\ M\dot{v} &= -Cv - K(x + y - x_0) + f_{ext}. \end{aligned}$$

When implemented on the GPU this correction term introduces only an extra texture.



```
fps: 60.026909
Iters: 4
Time Verlet: 0.001487
Time Constraints: 0.040877
Time InterCollisions: 0.018010
Max Els: -1
```

Figure 3: *Collision and Self-collision response example.*

5. Results

We have made a performance test between the three methods discussed in this paper: Mass-Spring, Constraint method and FME. We use a simple cloth model consisting in a rectangular geometry hanging for two points (see fig. 4). The results are shown in table 1 frame rates are obtained using a PC with an AMD 3Ghz and 7800 Nvidia GTX graphic card. As one can see, our FME implementation is still real time for the biggest test mesh. It obtains similar time performance for 64×64 mesh but with much more realisme due to the dynamic model.

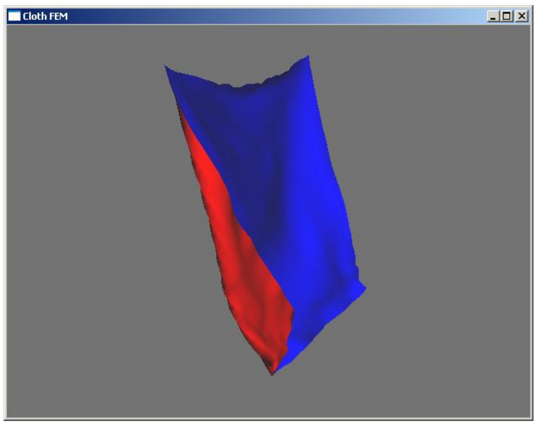


Figure 4: *FEM simulation corresponding to the test model used for the three methods.*

In figure 5 we show an exemple of a more complex cloth scene simulated with FEM on the GPU. Both, collisions and

Table 1: *Frame rates obtained with the three methods with CPU and GPU implemetations*

CPU (GPU)	Mass-Spring	Constraints	FEM
32×32	333 (724)	11 (80)	72 (648)
64×64	22 (181)	1.7 (14.4)	19 (171)
128×128	8 (60)	0.3 (2.6)	3 (27)

self-collisions with the woman body and the cloth respectively are detected and treated according to the method explained in the previous sections.

6. Conclusions

We have presented an optimal implementation of the FME method for cloth simulation on the GPU. The realism achieved by this approach is ensured by the model because it is based on engineering parameters. The fact that the method is appropriate for unstructured meshes gives more versatility for modelling garments or other cloth pieces.

Our complete simulation runs on the GPU, this include the solver for differential equations, the linear system solver and collisions and self-collisions. We have implemented also in the GPU two more methods in order to compare efficiency and visual results.

7. Acknowledgments

This research is partially supported by the TIN2004-08065-C02-01 CYCIT, and FISs 03/0102 grants. The authors want to thank Dr. Bernhard Spanlang for several interesting discussions about this topic.

References

[Bar96] Baraff D., Linear-Time Dynamics Using Lagrange Multipliers. Computer Graphics Proceedings, Annual Conference Series, 96: 137–146, 1996.

[BW98] Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. Computer Graphics Proceedings, Annual Conference Series, 98: 3–54, 1998. 2, 5

[BHW94] Breen, D., House, D. and Wozny, M. . Predicting the drape of woven cloth using interacting particles. Computer Graphics Proceedings, Annual Conference Series, 94: 365–372, 1994. 2

[CYT92] Carignan, M., Yang, Y., Thalmann, N. M., and Thalmann, D. Dressing animated synthetic actors with complex deformable clothes. Computer Graphics Proceedings, Annual Conference Series, 92: 99–104, (1992). 2

[CK02] K.-J. Choi and H.-S. Ko. Stable but Responsive Cloth. In Siggraph, pages 604-611. ACM Press, July 21-25 2002. 4

[CST02] Cordier F., Seo H., and Magnenat-Thalmann N., Made-to-Measure Technologies for an Online Clothing Store. IEEE Computer graphics and applications, 38–48, January 2002.

[EWS96] Eberhardt B., Weber A. and Strasser W. A fast, flexible, particle-system model for cloth draping. IEEE Computer Graphics and Applications, 16: 52–59, 1996. 2

[EG02] Etzmuss O. and Gross. J., Deriving a particle system from continuum mechanics for the animation of deformable objects. IEEE Trans. Vis. and Computer Graphics, 2002. 4

[EKS03] Etzmus O., Keckeisen M., Straser W., A Fast Finite Element Solution for Cloth Modelling. Proc. Pacific Conference on Computer Graphics and Applications-03, 244–251, 2003. 2, 3, 4

[GPU-2] Lefohn A., Kniss J. and Owens J. Implementing Efficient Parrallel Data Structures on GPUs. In GPU gems 2: Programming Techniques for High-Performance Graphics and General-Purposes Computation. Edited by Matt Pharr, pp. 512–545. Addison Wesley, 2005. 4

[GOL50] Goldstein H., Clasical Mechanics. Addinson-Wesley, 1950. 2

[Gr03] Green S., Nvidia developers 2003 http://developer.nvidia.com/object/demo-cloth-simulation.html 2

[HE01] Hauth M. and Etzmuss O. A high performance solver for the animation of deformable objects using advanced numericla methods. Computer Graphics Forum. Vol. 20 num. 3, 1–10, 2001. 2

[Jac01] Jacobsen T., Advanced Character Physics. Proc. Game Developers Conference'01, 1–10, 2001 2

[KJ01] Kolb A., John L., Volumetric model repair for virtual reality applications. In Eurographics short presentations 2001, Univ. of Manchester, 249–256, 2001. 2

[KLR04] Kolb A., Latta L. and Rezk-Salama C., Hardware-based simulation and collision detection for large particle systems. In proc. Graphics Hardware'04, T. Akenine-Mőller, M. McCool Ed., 1–9, 2004. 2

[KW03] J. Krüger, R. Westermann. Linear Algebra Operators for GPU Implementation of Numerical Algorithms. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003). 4

[Miralab] MiraLab, http://miralabwww.unige.ch/. 2

[MDM02] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, B. Cutler, Stable real-time deformations, Proc. of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, 49–54, 2002. 2, 3

[NG96] Ng H.N., Grimsdale R.L., Computer graphics techniques for modeling cloth. IEEE Computer Graphics and Applications 16: 28–41, 1996. 2

[NPF05] Nesme, M., Payan Y. and Faure, F., Efficient, Physically Plausible Finite Elements. In Proc. Eurographics (short papers), J. Dingliana and F. Ganovelli ed, 2005. 3

[PFT86] Press W.H., Flannery B.P., Teukolsky S.A., and Vetterling W.T. Numerical Recipes. Cambridge University Press, 1986.

[Pro95] Provot, X. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In Proceedings of Graphics Interface 95, 141–155, 1995. 1, 2

[RSS05] Rodríguez J., Sainz M., and Susin A., GPU Based Cloth Simulation with Moving Humanoids, Proc. Eurographics Spanish Chapter CEIG-05, J. Regincós, D. Martín. edit., 147-155, 2005. 5

[THM03] M. Teschner and B. Heidelberger and M. Mueller and D. Pomeranets and M. Gross, Optimized spatial hashing for collision detection of deformable objects. Proc. Vision Modeling, Visualitation, 47–54, 2003. 5

[THM04] Teschner M., Heidelberger B., Müller M., Gross M. A versatile and robust model for geometrically complex deformable solids, Proc. of the Computer Graphics International (CGI'04), 312–319, 2004. 2

[TPB87] Terzopoulos D., Platt J.C. and Barr A.H. Elastically deformable models. Computer Graphics (Proc. SIGGRAPH), 21: 205–214, 1987. 2

[TF88] Terzopoulos and K. Fleischer. Deformable models. Visual Computer, 4:306–331, 1988. 2

[VSC01] Vassilev, T., Spanlang, B., Chrysanthou, Y. Fast Cloth Animation on Walking Avatars. Computer Graphics Forum, vol 20 num. 3, 1–8, 2001. 2, 5

[VMT01] Volino P., Magnenat-Thalmann N. Comparing Efficiency of Integration Methods for Cloth Animation", Proceedings of Computer Graphics International (CGI), IEEE Press, pp. 265–274, Hong-Kong, July, 2001 Topic: Virtual Clothing Authors: P.Volino, N.Magnenat-Thalmann 2

[WBK01] Witkin A., Baraff D. and Kass M., Physically-based Modeling. SIGGRAPH Course notes. 2001.

[ZT93] Zienkiewicz O. and Taylor R., The Finite Element Method, 4th ed. McGraw-Hill, 1993. 3

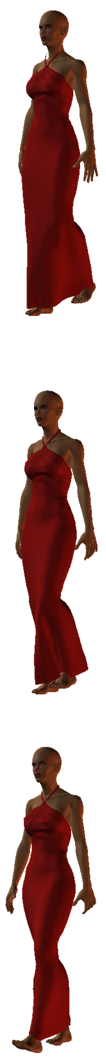


Figure 5: *Three frames of our FEM cloth simulation on the GPU corresponding to a moving woman wearing a night dress.*