Efficient Cloth Model for Dressing Animated Virtual People

Tzvetomir I Vassilev, Bernhard Spanlang
Department of Computer Science, University College London
Gower Street, London WC1E 6BT, United Kingdom
{t.vassilev|b.spanlang}@cs.ucl.ac.uk

Abstract

This paper describes a fast technique for dressing virtual humans with different pieces of clothing. The garments are constructed of cutting patterns seamed around the body. The system reads a body file and a cutting pattern file and produces a new dressed body file. The method exploits a mass-spring model of cloth but applies a new velocity modification approach to overcome its super-elasticity. This results in a more realistic cloth simulation with one more control parameter - stretching threshold. Unlike existing systems, which use object-space checks for collision detection, our approach exploits an image-space interference test. Hardware graphics accelerators are used not only for generating depth maps of the body contours but also to interpolate the normal vectors of the body. Responses to collisions are resolved modifying velocities of the cloth vertices and not applying penalty forces or energy fields as in other approaches. As the results section shows the system produces realistic images and makes it possible to sew a garment around a virtual human body in about a second on a mid-range PC.

Keywords: cloth simulation, physically based modelling, 3D scanning, image based collision detection

1 Introduction

The main objective of this work is to develop a technique for clothing virtual humans. The technique should be fast enough for an interactive system for dressing virtual people. The body model is acquired with a 3D scanner, so it represents very accurately the 3D body shape of a real person. The ultimate goal is to implement a system on the WEB, where customers will be able to upload the 3D virtual representation of their body, browse different types of clothes, try them on, and buy, if they are satisfied. Because of the accuracy of 3D scanning technology it will be possible not only to try on different types of clothes, but also to fit different sizes.

1.1 State of the art in body capturing technology

The fast development of 3D scanning technology evolved to whole body 3D scanners capable of capturing the 3D shape of the human body see Horiguchi (1998). Now there are several manufacturers on the market producing such scanners: Hamamatsu Photonics (Japan), Cyberware (USA), TelMat (France), Vitonic Viro 3D (Germany), etc. The output of the scanners is a cloud of 3D points, but using surface reconstruction software as in Dekker et al. (1998), a triangulated surface of the body can be obtained.

1.2 Previous work in cloth simulation

Physically based cloth modelling has been a problem of interest to computer graphics researchers for more than a decade. First steps, initiated by (Terzopoulos et al., 1987; Terzopoulos and Fleischer, 1988), characterised cloth simulation as a problem of deformable surfaces and used the finite element method and energy minimisation techniques borrowed from mechanical engineering. Since then other groups have been formed challenging the cloth simulation, using particle based

Breen et al. (1994), Eberhardt et al. (1996) or energy based approaches Carignan et al. (1992), Baraff and Witkin (1998). A more detailed survey on cloth modelling techniques can be found in the paper by Ng and Grimsdale (1996). Many of the approaches described above have a good degree of realism simulating the cloth but their common drawback is low speed. A relatively good result demonstrated by Baraff and Witkin (1998) is 14 seconds per frame for the simulation of a shirt with 6,450 nodes on a SGI R10000 processor. This is the main reason why these techniques cannot be applied to an interactive system that will work on Internet. Provot (1995) used a mass-spring model to describe rigid cloth behaviour, which proved to be much faster than the techniques described above. Its major drawback is the super-elasticity. In order to overcome this problem he applied a position modification algorithm to the ends of the over-elongated springs. However, if this operation has modified the positions of many vertices, it may have elongated other springs. That is why this approach is applicable only if the deformation is locally distributed, which is not the case when simulating garments on a virtual body.
Collision detection and response prove to be the bottleneck of dynamic simulation algorithms that use highly discretised surfaces. Most of the existing algorithms for detecting collisions between the cloth and other objects in the scene are based on geometrical object-space (OS) interference tests. Some apply prohibitive energy field around the colliding objects as in Terzopoulos et al. (1987), but most of them use geometric calculations to detect penetration between a cloth particle and a face of the object together with optimisation techniques in order to reduce the number of checks. The most common approaches are voxel or octree subdivision as described in Glassner (1998). Another solution is to use a bounding box hierarchy. See Baraff and Witkin (1998) or Provot (1995). Objects are grouped hierarchically according to proximity rules and a bounding box is pre-computed for each object. The collision detection is then performed analysing bounding box intersections in the hierarchy. Other techniques exploit proximity tracking as shown in Pascal and Magnenat-Thalmann (1995) or curvature computation as in Provot (1995) to reduce the big number of collision checks, excluding objects or parts which are impossible to collide. Recently techniques based on image-space (IS) tests were developed. Algorithms as described in Shinya and Forgue (1991), Rossignac et al. (1992), Myszkowski et al. (1998) and Baciu et al. (1999) use the graphics acceleration hardware to render the scene and then perform checks for interference between objects based on the depth map of the image. In this way the 3D problem is reduced to 2.5D. As a result of using the graphics hardware these approaches are very efficient.

1.3 Contribution of this work

Our method exploits a mass-spring cloth model but we introduce a new approach to overcome its super-elasticity. Instead of modifying the positions of end points of the springs that were already over-elongated, the algorithm checks their length after each iteration and does not allow elongation more than a certain threshold modifying velocities of the corresponding vertices. Most of the existing techniques Terzopoulos et al. (1987), Baraff and Witkin (1998) resolve collisions applying repulsing or penalty forces, which makes the resulting system of equations complex and time-consuming to solve.
To detect collisions we use an image space approach exploiting the graphics hardware. Image space approaches are scalable which means the number of vertices of the body does not affect the performance of the simulation. We describe two methods to respond to detected collisions. The first method is for static bodies. In this case we modify velocities of the cloth vertices. And the second method treats animated bodies. Here we change the forces of the cloth vertices. As a result of using only velocity and force modification for coping with the super-elasticity and responding to collisions, our algorithm is fast and allows a very fast draping simulation. For example, the time for the simulation of a shirt of 1800 vertices draping on a human body of 5520 vertices is just about 15 ms per iteration on a SGI with R12000 processor. Animating and rendering of a dressed body can be achieved in less then a second per frame. The rest of the paper is organised as follows. The next section outlines the cloth model and its integration over time. Section 3 describes the technique for constraining the super-elasticity, while sections 4 and 5 are dedicated to dealing with collisions. Section 6 gives more details on the way of dressing a human body. Section 7 shows

experimental results and section 8 outlines future work.

2 Mass-spring model of cloth

2.1 Topology

The elastic model of cloth is a mesh of $l \times n$ mass points, each of them being linked to its neighbours by massless springs of natural length greater than zero. There are three different types of springs:

- Springs linking vertices $[i, j]$ with $[i + 1, j]$, and $[i, j]$ with $[i, j + 1]$ are called "structural springs";
- Springs linking vertices $[i, j]$ with $[i + 1, j + 1]$, and $[i + 1, j]$ with $[i, j + 1]$ are called "shear springs";
- Springs linking vertices $[i, j]$ with $[i+2, j]$, and $[i, j]$ with $[i, j+2]$ are called "flexion springs".

As one can guess from the names the first type implements resistance to stretching, the second - resistance to shearing and the third - resistance to bending.

2.2 Forces

Let $\mathbf{p}_{ij}(t)$, $\mathbf{v}_{ij}(t)$, $\mathbf{a}_{ij}(t)$, where i=1...l and j=1...n, be correspondingly the positions, velocities, and accelerations of the mass points at time t. The system is governed by the basic Newton's law:

$$\mathbf{f}_{ij} = m\mathbf{a}_{ij} \tag{1}$$

where m is the mass of each point and f_{ij} is the sum of all forces applied at point p_{ij}. The force f_{ij} can be divided in two categories. The **internal forces** are due to the tensions of the springs. The internal force applied at the point p_{ij} is a result of the stiffness of all springs linking this point to its neighbours:

$$\mathbf{f}_{int}(\mathbf{p}_{ij}) = -\sum_{k,l} k_{ijkl} \left(\mathbf{p}_{kl}\vec{\mathbf{p}}_{ij} - l^0_{ijkl} \frac{\mathbf{p}_{kl}\vec{\mathbf{p}}_{ij}}{|\mathbf{p}_{kl}\vec{\mathbf{p}}_{ij}|} \right) \tag{2}$$

where k_{ijkl} is the stiffness of the spring linking p_{ij} and p_{kl} and l^0_{ijkl} is the natural length of the same spring. The **external forces** can differ in nature depending on what type of simulation we wish to model. The most frequent ones will be: Gravity: $\mathbf{f}_{gr}(\mathbf{p}_{ij}) = m\mathbf{g}$, where $\mathbf{g}$ is the gravity acceleration; Viscous damping: $\mathbf{f}_{vd}(\mathbf{p}_{ij}) = -C_{vd}\mathbf{v}_{ij}$, where C_{vd} is a damping coefficient. For more information on how to model wind see Provot (1995). All the above formulations makes it possible to compute the force $\mathbf{f}_{ij}(t)$ applied on point $\mathbf{p}_{ij}$ at any time t. The fundamental equations of Newtonian dynamics can be integrated over time by a simple Euler method:

$$\left| \begin{array}{l} \mathbf{a}_{ij}(t + \Delta t) = \frac{1}{m}\mathbf{f}_{ij}(t), \\ \mathbf{v}_{ij}(t + \Delta t) = \mathbf{v}_{ij}(t) + \Delta t\mathbf{a}_{ij}(t + \Delta t), \\ \mathbf{p}_{ij}(t + \Delta t) = \mathbf{p}_{ij}(t) + \Delta t\mathbf{a}_{ij}(t + \Delta t) \end{array} \right. \tag{3}$$

where Δt is the chosen time step. More complicated integration methods, such as Runge-Kutta Press et al. (1992), can be applied to solve the differential equations. This, however, reduces the speed significantly, which is very important in our case. The Euler eq.3 are known to be very fast and give good results, when the time step Δt is less than the natural period of the system $T_0 \approx \pi\sqrt{\frac{m}{K}}$. In fact our experiments showed that the numerical solving of eq.3 is stable when

$$\Delta t \leq 0.4\pi\sqrt{\frac{m}{K}}, \tag{4}$$

where K is the highest stiffness in the system.

3 Constraining super-elasticity

The major drawback of the mass-spring cloth model is that it is very elastic. This means that the deformation rate for some springs is too high and as a result the cloth stretches under its own weight, something that does not happen to real cloth.

3.1 Increasing stiffness

One way to avoid the super-elastic effect is to increase the stiffness of the springs. For the same external forces (in our case gravity and damping) the elongation rate should be lower for stiffer springs. This approach, however, has an obvious drawback. Eq. 4 shows that increasing the stiffness coefficient K will require decreasing the time step Δt. This means that for the same animation time the number of iterations will be greater and the algorithm will be more time consuming. Another drawback is that even if the stiffness is increased, the resulting material behaves more like rubber rather than cloth.

3.2 Position modification

Provot (1995) proposed to cope with the super-elastic effect using position modification. His algorithm checks the length of each spring after each iteration and modifies the positions of the ends of the spring, if it exceeds its natural length with more than a certain value (10% for example). This modification will adjust the length of some springs but it might over-elongate others. So, the convergence properties of this technique are not clear. It proved to work for locally distributed deformations but no tests were conducted for global elongation.

3.3 Velocity modification

The main problem of the position modification approach is that first it allows the springs to over-elongate and then tries to adjust their length modifying positions. This of course is not always possible because of the many links between the mass points. Our idea was to find out a constraint that does not allow any over-elongation of springs. The algorithm works as follows. After each iteration it checks for each spring whether it exceeds its natural length by a pre-given threshold. If this is the case, the velocities are modified, so that further elongation is not allowed. The threshold value usually varies from 1% to 10% depending on the type of cloth we want to simulate. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the positions of the end points of a spring found as over-elongated, and $\mathbf{v}_1$ and $\mathbf{v}_2$ be their corresponding velocities (see fig. 1). The velocities $\mathbf{v}_1$ and $\mathbf{v}_2$ are split in two components $\mathbf{v}_{1t}$ and $\mathbf{v}_{2t}$, along the line connecting $\mathbf{p}_1$ and $\mathbf{p}_2$, and $\mathbf{v}_{1n}$ and $\mathbf{v}_{2n}$, perpendicular to this line. Obviously the components causing the spring to stretch are $\mathbf{v}_{1t}$ and $\mathbf{v}_{2t}$, so the algorithm sets them to zeros. If this is applied to all strings, the stretching components of the velocities are removed and in this way further stretching of the cloth is not allowed. As the results show this approach works fine not only for local but also for global deformations.

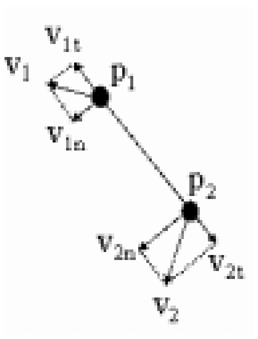

Figure 1: Velocity modification for over-elongated springs

Collision detection algorithm	Number of Iterations to sew a shirt	Total time ins sec	Time per iteration in ms
IS	82	1.3	15.8
OS	149	4.6	30.8

Table 1: Comparison between traditional OS approach to the IS approach for a shirt of 1800 vertices

4 Collision detection

To compare different collision detection approaches we implemented two methods. An object space (OS) technique, which uses bounding boxes for increasing performance, and an image space (IS) method using depth-buffer and frame-buffer of the graphics acceleration hardware.

4.1 Object Space

Our OS algorithm checks for collisions between each point of the cloth and each face of other objects in the scene. In our case this is the customer's scanned body, but it could also be any other rigid object. In order to avoid a big number of $O(n^2)$ comparisons, first we build a quad tree of bounding boxes for each solid object on which the cloth drapes. The leaves of the tree are the faces of the body surface with their normal vectors. A quad tree was implemented for the following reason. After processing the 3D scanner raw data the human body is segmented in six parts: head, torso, two arms and two legs. The surface of each body part is presented as a rectangular topology mesh (quad-mesh) of $m \times n$ vertices, where m is the number of horizontal body slices and n is the number of points within a slice. Subdivision in each of the two directions gives the natural quad structure. One such tree is built for each body part. The recursive search of the tree leads to $O(n \log_4 n)$ comparisons. For a static rigid body the tree and the normals for each face are computed in advance, which speeds up the algorithm.

4.2 Image Space

In the IS approach the scanned body is rendered without clothes from both the front and the back. The body is painted with the colour representing the appropriate body normals. This is done by setting the colour of each vertex (R, G, B) to the value of the vertex normal (x, y, z). See fig. 3. The rendering processes generate two depth buffers and two frame buffers for the front and the back of the scanned body. See fig. 2 and 3. To check for a collision we convert the appropriate vertex coordinates of the garment to an index in the depth and normal buffer. A check for collision is accomplished by simply comparing the z value of the vertex coordinate with the corresponding value in the depth buffer. Apparently this can be achieved in $O(1)$ time. The normals encoded in (R, G, B) values are automatically interpolated by the graphics acceleration hardware and can be obtained from the normal buffer using the appropriate map-index. We pre-compute the depth and normal maps to increase the performance of the actual cloth simulation.

4.3 Comparison

Table 1 shows the results of comparing IS and OS techniques when sewing a shirt composed of 1800 vertices. The IS approach converges much faster, not only because its time per iteration is shorter, but it also produces a smaller number of iterations.

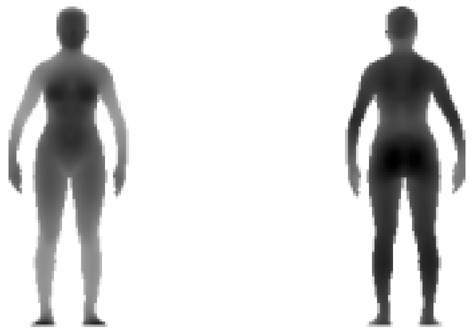

Figure 2: depth buffers of the front and of the back

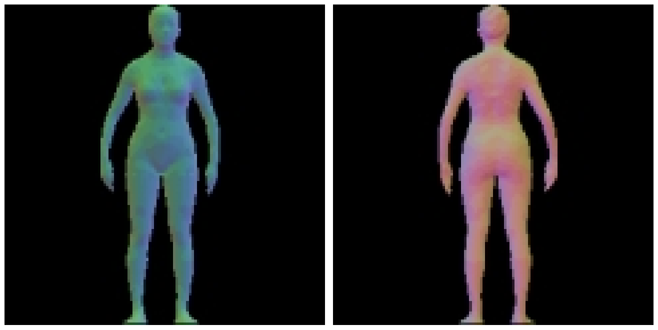

Figure 3: normal buffer of the front and of the back

5 Resolving collisions

After a collision was detected, the algorithm should compute a proper response of the whole system. In the static case our approach does not introduce additional penalty, gravitational or spring forces; it just manipulates the velocities as in Moore and Wilhelms (1988), which proved to be very efficient. Let $\mathbf{v}$ be the velocity of the point $\mathbf{p}$ colliding with the surface $\mathbf{s}$ (fig. 4). The surface normal at the point of collision is denoted by $\mathbf{n}$. If $\mathbf{v}_t$ and $\mathbf{v}_n$ are the tangent and normal components of $\mathbf{v}$, then the resultant velocity can be computed as:

$$\mathbf{v}_{res} = C_{fric}\mathbf{v}_t - C_{refl}\mathbf{v}_n \tag{5}$$

where C_{fric} and C_{refl} are a friction and a reflection coefficients, which depend on the material of the colliding objects. Unfortunately this approach is difficult to apply when dealing with animated bodies. In the animated case it is easier to change the forces of the cloth vertices proportional to the distance to the surface and along the corresponding body surface normals.

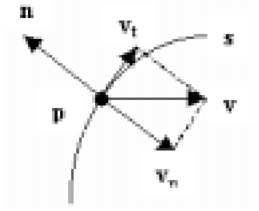

Figure 4: Resolving collisions manipulating velocities

6 Dressing the human body

Once a cloth model and algorithms for collision detection and response have been implemented, the only thing needed to complete the system for dressing virtual humans is cutting patterns. Our software reads an input body file and a garment text file, describing the number of cutting patterns and necessary seaming information, and produces an output file of a dressed human body. If the body presented in the input file is already dressed, then the existing clothing is considered as a solid object in the scene, i.e. it participates in the collision detection process. In this way it is possible to put on several layers or several products as shown in figure 8 .

6.1 Cutting pattern

The ideal solution would be to read output text files from the existing clothing CAD/CAM systems such as GERBER. They export DXF files describing the cutting pattern geometry and seaming information. Our current version reads DXF files containing cutting pattern geometry. Since there is no agreed standard for seaming information we had to add this by hand. A standard for describing seaming information is in development.

6.2 Meshing the patterns

As explained in the previous sections our cloth model is a rectangular topology mesh of $l \times n$ mass points connected with springs. So, in order to apply the algorithm, we had to implement meshing software that reads the cutting patterns and produces the rectangular topology as shown on figure 5. After the cutting patterns have been produced and meshed, they are positioned around the body and elastic forces are applied along the seaming lines. After a certain number of iterations the patterns are seamed, i.e. the shirt is 'put on' the human body. Then several more iterations are performed applying gravity in order to drape the cloth.

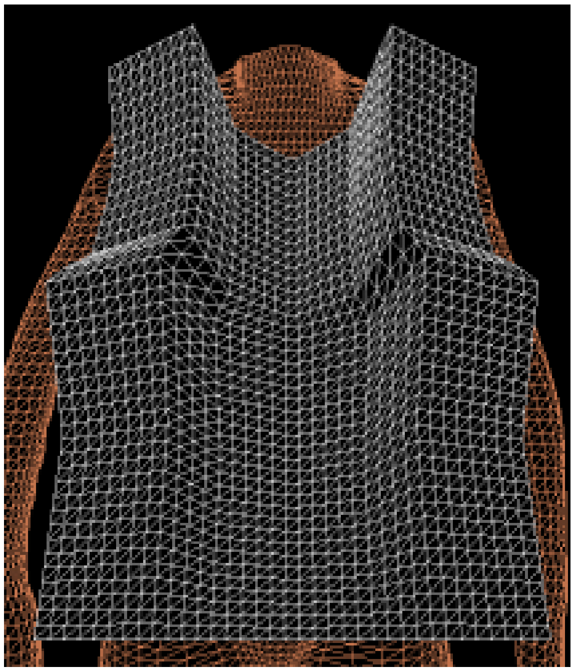

Figure 5: Cutting pattern of a shirt with a rectangular topology mesh

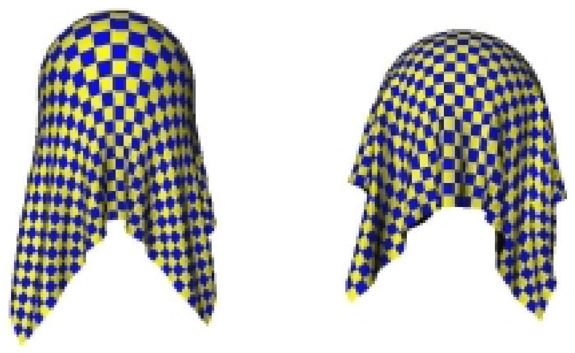

Figure 6: Left: original elastic model; right: model with velocity modification

7 Results

The algorithms were implemented on a SGI workstation with a R12000 processor using Open Inventor and OpenGL libraries for rendering the images. Fig. 7 shows the draping of a rectangular piece of cloth over a sphere using the original elastic model with no restriction to stretching (left) and with the velocity modification approach described above with a 5% elongation threshold (right). As one can see the original cloth model stretches over its own weight, which is never the case with real cloth. The model with velocity modification overcomes this problem.

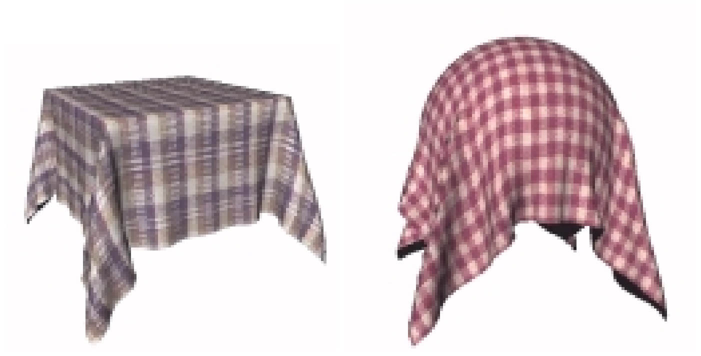

Figure 7: Draping the cloth over simple objects applying texture

Fig. 7 shows rectangular pieces of cloth draping over simple objects: table and sphere. Incorporating textures makes images very realistic. Fig. 8 exhibits dressed human bodies with different pieces of clothing. Fig. 9 shows an animated dressed woman. The bodies were acquired at University College London using a Hamamatsu Photonics whole body 3D scanner. The dress, the skirt and the sleeveless shirt were built out of two cloth patches, the trousers - out of four and one patch was needed for each sleeve of the female shirt. An elasticity threshold of 5% was set for the cloth model. The numbers of vertices on the cloth are listed in Table 2, while the body consists of 5520 vertices. Table 2 indicates the times necessary to dress a human body with different garments. The numbers of vertices were the minimum needed to produce visually pleasing results. Smaller number results in 'holes' on the cloth, that is the collision detection does not work properly because of the very rough mesh. The table does not include the set-up times necessary to read the files, generate the bounding box trees, etc.

Product	Number of cloth vertices	Number of iterations	Total time in sec	Time per iteration in ms
skirt	1152	80	1.25	15.6
shirt	1800	149	4.4	29.5
trousers	2048	147	4.7	31.9
dress	3840	150	7.8	52.0

Table 2: Times necessary to dress a virtual body with the listed products

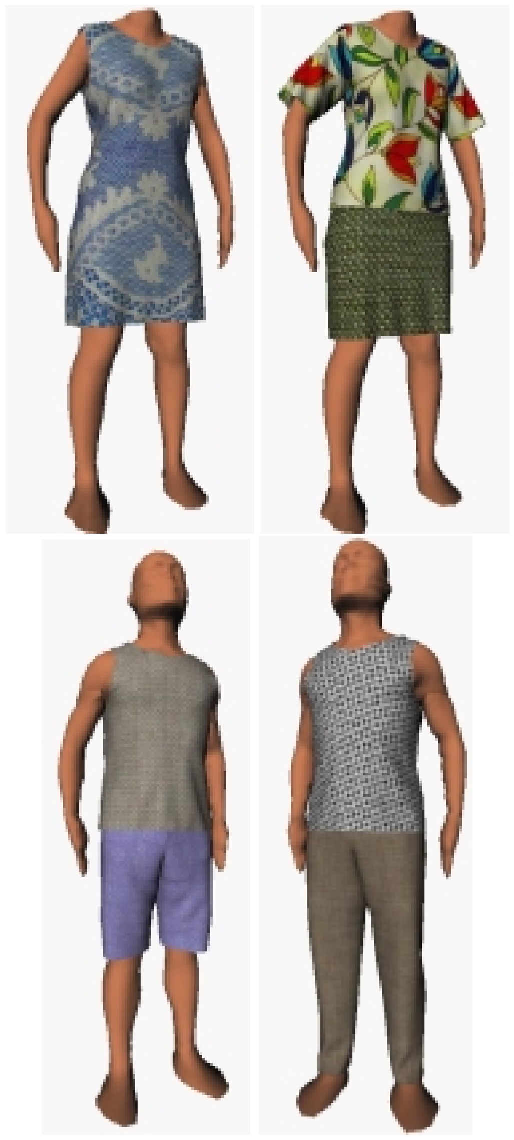
Figure 8: Dressing virtual people with different pieces of clothing

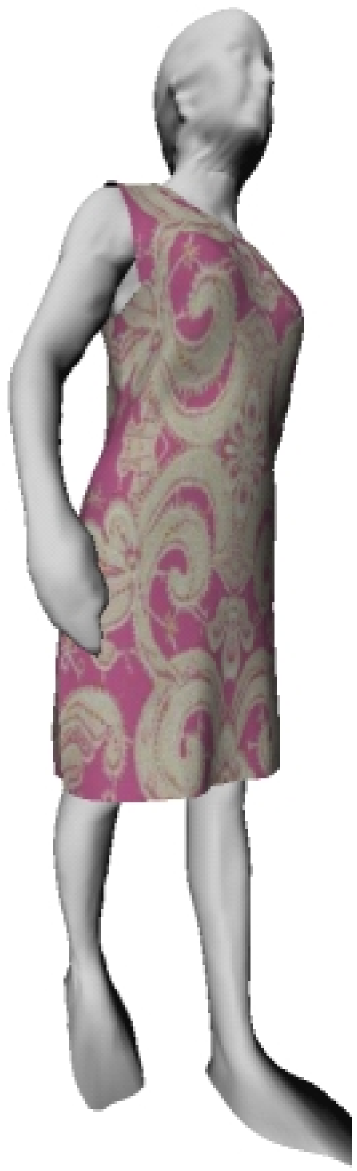

Figure 9: Example of a dressed animated woman

Fig. 10 illustrates how computational time changes with increasing the number of vertices on the cloth. As one can see, the dependence is linear, which indicates the low $O(n)$ complexity of the algorithm.

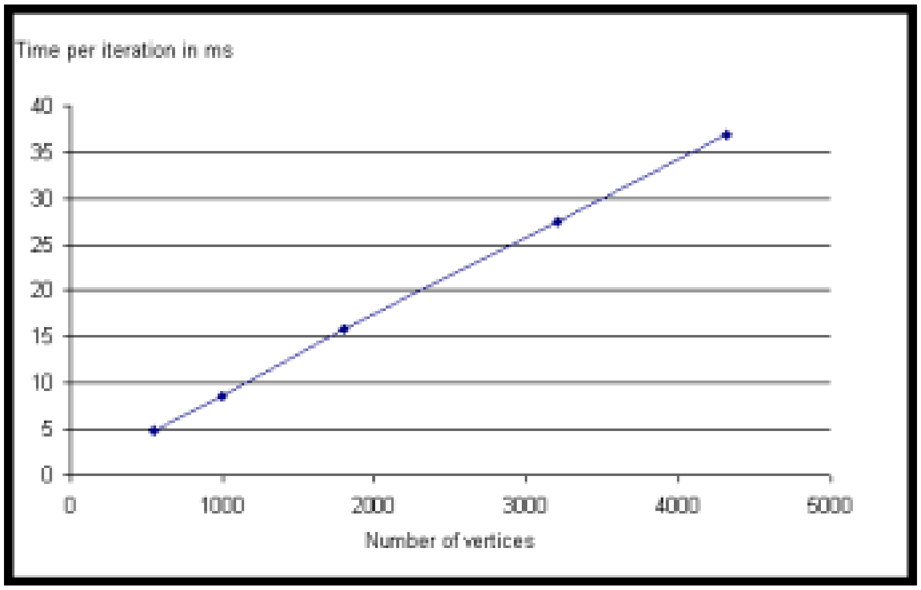

Figure 10: Time per iteration versus number of vertices on the cloth

8 Conclusions and Future work

An efficient technique for cloth simulation has been presented. It uses a mass-spring model with velocity and force modification methods for coping with the super-elasticity and for resolving responses to collisions. As a result the speed is good enough for almost real time simulation of draping virtual garments on animated virtual humans. The body surface is acquired through a 3D scanner, which makes the approach very accurate and allows users to try on different sizes of clothing and see how they fit them. So far we conducted experiments on static and very crudely animated human bodies. Our future plans are to implement dynamic simulation of garments on naturally animated virtual actors using the presented approach.Animation will be done by applying motion tracker data and skin deformation techniques to the static body. We will evaluate the efficiency and compare it to existing techniques.

9 Acknowledgements

This work is a part of the Foresight LINK project "3D Centre for Electronic Commerce" at the University College London. Many thanks to Laura Dekker and Ioannis Douros for supplying the 3D body models and to Prof. Philip Treleaven and Yiorgos Chrysanthou for their help and ideas.

References

Baciu, G., Wong, W. S., and Sun, H. (1999). Recode: an image-based collision detection algorithm. *The Journal of Visualization and Computer Animation*, 10(4):181–192.

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series*, pages 43–54. SIGGRAPH.

Breen, D., House, D., and Wozny, M. (1994). Predicting the drape of woven cloth using interacting particles. In *Computer Graphics Proceedings, Annual Conference Series*, volume 94, pages 365–372.

Carignan, M., Yang, Y., Thalmann, N. M., and Thalmann, D. (1992). Dressing animated synthetic actors with complex deformable clothes. In *Computer Graphics Proceedings, Annual Conference Series*, volume 92, pages 99–104.

Dekker, L., Douros, I., Buxton, B. F., and Treleaven, P. (1998). Building symbolic information for 3d human body modelling from range data. In *Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling*, pages 388–397.

Eberhardt, B., Weber, A., and Strasser, W. (1996). A fast, flexible, particle-system model for cloth draping. *j-IEEE-CGA*, 16(5):52–59.

Glassner, N. I. B. A. S. (1998). 3d object modelling. *SIGGRAPH*, 12(4):1–14.

Horiguchi, C. (1998). Hamamatsu, bl (body line) scanner. *International Archives of Photogrammetry and Remote Sensing*, XXXII(5):28–41.

Moore, M. and Wilhelms, J. (1988). Collision detection and response for computer animation. In *Computer Graphics Proceedings, Annual Conference Series*, pages 289–298. SIGGRAPH.

Myszkowski, K., Okunev, O., and Kunii, T. (1998). 3d object modelling. *The Visual Computer*, 11(9):497–512.

Ng, N. and Grimsdale, R. (1996). Computer graphics techniques for modelling cloth. *IEEE Computer Graphics and Applications*, 16:28–41.

Pascal, V. and Magnenat-Thalmann, N. (1995). Collision and self-collision detection: efficient and robust solution for highly deformable surfaces. *Sixth Eurographics Workshop on Animation and Simulation*, pages 55–65.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C, 2nd. edition.* Cambridge University Press.

Provot, X. (1995). Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Proceedings of Graphics Interface*, pages 141–155.

Rossignac, J., Megahed, A., and Schneider, B. O. (1992). Interactive inspection of solids: cross-section and interferences. *SIGGRAPH*, 26(2):353–360.

Shinya, M. and Forgue, M. C. (1991). Interference detection through rasterization. *j-J-VIS-COMP-ANIMATION*, 2(4):132–134.

Terzopoulos, D. and Fleischer, K. (1988). Deformable models. *The Visual Computer*, 4(6):306–331.

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. *Computer Graphics (Proc. SIGGRAPH'87)*, 21(4):205–214.