REAL-TIME CLOTHING: GEOMETRY AND PHYSICS

Isaac Rudomín **José Luis Castillo**
Department of Computer Science, ITESM-CEM
3.5 Carretera Lago de Guadalupe, Atizapán, Estado de México , México
rudomin@campus.cem.itesm.mx

ABSTRACT

This paper describes a technique for animating in real time garments placed over an articulated character. In order to do this, the character to be dressed must be approximated using a hierarchy of ellipsoids. The pieces of clothing are represented using mass-spring particle systems. First the particles move following the ellipsoids; this is followed by the application of dynamic forces. Finally, penetration of the character's ellipsoids by any particle is corrected. This method has been optimised and is fast enough to deliver real-time performance on mid-range PCs and workstations, using only portable and standard C++ and OpenGL code.

Keywords: cloth simulation, real-time clothing.

1. INTRODUCTION

Cloth modelling and simulation has always been a great challenge inside the computer graphics field. Existing approaches can usually be classified as geometric, physically based, or a combination of both (hybrid). The simplest physically-based method uses mass-spring particle systems to simulate the topology and behaviour of cloth. The main advantage of using such a system is that it can effectively simulate the movement of cloth and it is relatively easy to implement.

Physical simulation of cloth in real time has to meet several requirements that greatly limit the development or even the existence of such systems. Among these we could mention stability, robustness and, of course, speed (see [Baraff98, Desbrun99, Volino00]):

1. Stability is needed to deal with the fact that whatever the conditions presented during the simulation the system must behave in a correct and predictable manner. Unnatural or erratic cloth appearance and movement must be avoided at all costs.
2. Robustness is an even more difficult requirement, especially when the cloth is placed in an interactive environment, i.e. a virtual reality application or a video game, where interaction or the fast-changing conditions must be handled properly.
3. Speed is obviously the most important aspect of any real time system. There are two major obstacles that make these requirements difficult to fulfil: the cloth model itself and the detection of collisions. Most highly accurate physically-based cloth simulation systems are just too slow to be used in an interactive or real time applications. They are computationally heavy and difficult to optimise to a level that would permit them to be significantly accelerated.

On the other hand, over-simplified physically-based cloth models have problems of their own. They usually present strange or unnatural behaviour, like super-elasticity or high-compression. Baraff et al. [Baraff98], define an implicit integration scheme that allows the system to take larger step in the simulation and to include hard constraints and a broad model. However, it is more suitable for animation or off-line simulation than for interactive applications.

Collision detection calculations can prevent the application from reaching real time performance. Natural-looking clothing can only be achieved if the way garments are placed over other surfaces is consistent with the shape of those surfaces and with the way gravity and other forces act on the clothes, and this effect strongly depends on how the collisions are handled. Thus, detecting and responding properly to collisions can be a time-consuming task, especially when the cloth covers a constantly moving object, as is the case with clothing dressing an articulated character.

Several research projects have tried to animate cloth in real time. Desbrun et al. [Desbrun99] use a simplified version of Baraff's implicit integration method and effectively achieves interactive rates on a high-end graphics computer. Other approaches take advantage of available graphics hardware to accelerate certain operations. Vassilev et al. [Vassilev01] propose a fast method for dressing human characters that bases its performance on the use of image-space operations (as opposed to object-space) for collision detection and normal calculation. Oshita [Oshita01] represents cloth as a sparse triangle mesh. Particle positions are calculated with this sparse mesh, and interpolation is performed to generate a dense mesh. A technique called PN-triangles is used (see [Vlachos01]). Animating only a few particles is much faster and PN-triangles can be created automatically by certain graphics hardware.

Other approaches have been suggested; the most relevant to this paper is the one described in [Perez99] and [Rudomin00]. In their approach, a hybrid geometry-physics method, a group of implicit ellipsoids are defined that approximate the shape of a human character. A scalar field is generated from those ellipsoids, and it is evaluated for every point of the cloth, after which they are moved accordingly. Since a different scalar field may be used for every

piece of garment, no collision detection calculations are needed between them, allowing the use of several layers of clothing. In this paper, we have modified this method with the intention of making it perform at interactive rates (our goal was at least 10 fps).

2. OUR SYSTEM

As just mentioned, our purpose is to extend the work done by Rudomín and Pérez-Urbiola in order to achieve real time performance. One of the main optimisations done is the shift from calculating implicit isosurfaces to ellipsoid distance calculations. This is much faster. In general, our system works as follows:

1. the character to be dressed is approximated using groups of ellipsoids;
2. after that, a triangle mesh representing clothing is placed over the character (where every vertex is taken as a particle and every edge as a spring),
3. then the particles of the mesh are moved with the ellipsoids.
4. subsequently, they are allowed to move freely using physics,
5. but kept always from penetrating any ellipsoid.

The ellipsoid approximation is used since it greatly simplifies the collision detection problem and makes it very fast to compute.

One of the main advantages of this system is that it works extremely fast even on mid-range computers. What we did was to optimise crucial and usually intensive operations in the simulation. These operations can be classified in four important groups:

1. the simplification of the character by using ellipsoids for the purpose of collision detection,
2. the use of a simple mass-spring system with first-order integration within a hybrid scheme,
3. the optimisation of the collision detection mechanism
4. the grouping of ellipsoids connected by an adjacency graph;

All of these are explained in detail in the following sections.

We intend to apply our system on virtual reality applications and video games, where interaction and stability is more important than accuracy.

2.1 Character Description and Cloth Models

Since our method is based on computing point-to-ellipsoid distances, the character to be dressed must first be approximated using a hierarchy of ellipsoids. These ellipsoids must closely match the character's shape and its hierarchy must be arranged in the exact same way as the character's hierarchy.

This allows animating the ellipsoids in the same manner as the character.

Ellipsoids are formed using scaled spheres, with a certain rotation and potentially different values for every dimension. This representation greatly enhances the speed at which penetration tests are performed in our system.

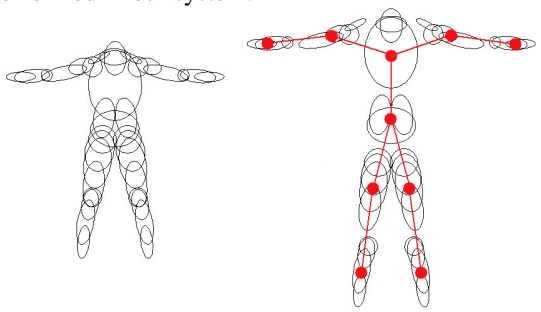


Figure 1 *An ellipsoid arrangement and adjacency graph*

Given that it is not efficient to verify every single ellipsoid for penetration, ellipsoids are organised in groups, with every group containing only a few ellipsoids, and then specifying which groups are connected. This is done through the use of a simple adjacency graph, represented by a matrix that indicates the connections between groups. For an example of the ellipsoid arrangement of a character, see Figure 1. The benefits derived from using such group organization of ellipsoids are discussed in section 2.4.

The garments are based on a mass-spring particle system. For the actual geometric model we can use any file describing a triangle mesh, and in fact, it can be any model generated by a modelling package, even with materials and textures applied. Every vertex in the geometric model is used as a particle and every edge in the mesh is considered a spring. The rest lengths of the springs are taken from the initial position of the cloth described in the file.

The shape and position of the cloth pieces must match the shape and position of the character to be dressed, or to be precise, the ellipsoids approximating it. For our tests, we used models generated with *MayaCloth*, a cloth-modelling program inside the *Alias/Wavefront Maya* animation package.

2.2 System Integration Schemes for Real Time Cloth Simulation

A mass-spring particle system consists of a set of particles with a certain mass and springs connecting them. These springs have associated rest lengths and depending on their lengths adopted during the simulation, they apply forces to the particles in order to restore their original state.

If the springs are stiff enough, they effectively help to maintain the original shape of the system. Other types of forces like gravity or wind may also be used to affect the particles. For the simulation of cloth, the particles are used to determine the

positions of vertices in a polygon mesh, usually a triangle or quad mesh. The springs may correspond to the edges of the polygons forming the mesh, but this is not necessary, and there can even be more springs than edges in the mesh, depending on the type of behaviour wanted.

Since a mass-spring system used for cloth simulation is a dynamic model, the type of system integration used is very important. This is especially true when interactive rates are needed.

There are basically two kinds of integration methods that are relevant: explicit and implicit. Explicit integration schemes, like the Euler or Runge-Kutta methods, are fast and easy to implement, but a small time step is needed in order to maintain system stability and congruence. On the other hand, an implicit integration method allows the use of much larger time steps but requires a lot more computations to resolve the system in every step. More over, as mentioned in [Volino00], a high-order explicit method like fourth-order Runge-Kutta is not suitable either for such an application if the system is too irregular.

Trying to approximate a rapidly changing system this way may lead to unpredictable results and instability. Consequently, for interactive or real time applications, where the time step is small and the regularity of the system cannot be determined, an explicit first or second order integration method would be a good choice. Depending on the actual time step, one could choose from explicit Euler or second-order Runge-Kutta. For our system, after extensive testing, we found that we really could not find a difference, and that since explicit first-order Euler was a little bit faster, we could get away with using it. Therefore, we chose the explicit Euler scheme; the cloth remains stable if (it is not too rigid) and using it saves some computation.

We think that this stability is due to the hybrid nature of our animation scheme, since we are moving the particles mostly by using geometric criteria, and only partially with the physical simulation.

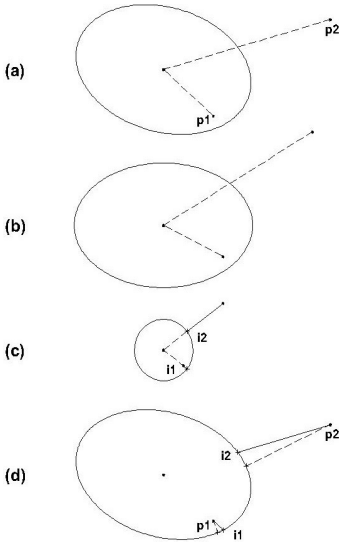
2.3 Efficient Distance Computations for Collision Avoidance

One of the main optimisations done in our system is the use of an efficient point-to-ellipsoid distance calculation. Actually, it is used for obtaining three different results:

1 The distance from the point to the ellipsoid;
2 Whether the point is inside the ellipsoid or not; and
3 The point on the surface of the ellipsoid where the intersection was found.

The intersection point mentioned above is not the closest point on the ellipsoid (see figure 2) and, consequently, the distance obtained is not the shortest either, but it is a close and fast approximation and has served its purpose well. To obtain this information easily and quickly, certain assumptions are made.

1. The ellipsoids are scaled spheres, with potentially different radiuses for every axis.
2. The matrices that transform every axis-aligned ellipsoid from the origin to their global positions and rotations are stored, as well as their corresponding inverse matrices.
3. Since the ellipsoids originally have only their positions relative to its parent, the direct and inverse transformation matrices are obtained by applying the complete ellipsoid hierarchy, saving the corresponding OpenGL ModelView matrices and calculating its inverses, see [Woo99, Angel00].
4. This is recalculated every time a certain ellipsoid is rotated or moved, but only the altered ellipsoid and its children are redone.



(a) two example points near an ellipsoid;
(b) their new positions after origin-centring and axis-aligning;
(c) their new positions after scaling, where penetration and points of contact are determined; and
(d) the intersection points found, after all the direct transformations.

Figure 2 *Procedure for testing for penetration of an ellipsoid and distance calculation*

2.4 Ellipsoid Group and Adjacency Graph

At the start of the simulation, every cloth particle is assigned to an ellipsoid, choosing the one that is closest to it based on the distances to all the ellipsoids.

This ellipsoid becomes its "parent ellipsoid". However, it is a dynamic assignation, since the parent ellipsoid may change during the simulation.

Based on this information, when an ellipsoid moves, its *children particles* (those that lie within a certain distance) move along with it. This facilitates the overall animation and avoids artifacts related to cloth penetration of fast-moving parts of the character. For every frame in the simulation, forces like gravity, wind and internal damping are applied to the particles.

Spring forces are also calculated and applied. After all the dynamic forces are accumulated, an

explicit integration is performed and the velocity and position of every particle is obtained. Once the new position for a particle is acquired, it is verified that the particle does not penetrate any ellipsoid. If this happens, the point closest to the penetrated ellipsoid is calculated as described above, and the particle is moved to that position.

The new information on this section is that when testing for penetration, distances are checked only against ellipsoids in the same group of the parent and in groups adjacent to it. The parent ellipsoid is re-assigned to the closest one calculated (note that this may in fact cause the particle to change from one group to another). This technique significantly reduces the number of checks made per particle/vertex, and allows the dynamic assignation of groups, with almost no computational penalty.

3. RESULTS

We analysed the performance of the system on several computers:

PC1 (Nbook)	PIII/750	S3 savage/8MB
PC2 (Desktop)	PIII/800	GloriaDCC/64MB
SGI Octane2	R1200/360	Oddysey/32MB

Table 1 shows the simulation (clothing adjustment) time in seconds, obtained by running the simulation with different options (blouse, pants or both blouse and pants). From this table we can conclude that simulation time allows us real time calculation.

Objects and number of vertices	PC 1	PC 2	SGI
Blouse (911)	0.008	0.003	0.010
Pants (944)	0.009	0.004	0.011
Both (1855)	0.017	0.007	0.021

Table 1 *Clothing adjustment time (in seconds)*

In all cases, we obtain frame-rates that are better than 30 fps, and sometimes up to 60 fps. This is consistent with and in fact exceeds our goal of obtaining simulation and rendering within interactive rates. In figure 3 we show the pants and blouse draped over a model in a sequence of different positions.

Figure 3 *Behaviour of clothing in real time*

4. CONCLUSIONS

The results show that this method is fast, stable and portable. As opposed to other approaches, this one performs well even on mid-range PCs, and does not rely on vendor-specific or expensive hardware. Partial OpenGL hardware acceleration is sufficient. The method is suitable for real time applications, where interaction is constant and accuracy is not the main aspect of the simulation.

Future work:

1. Extend the method to support several layers of garments while retaining real-time performance;
2. Use fewer particles, and interpolating a dense mesh by hardware or software;
3. Improve the quality of the physics simulation; Avoid self collision of the garments;
4. Use hardware vertex-shading techniques to automatically move the cloth vertices;
5. Add wrinkles and other details and accessories for a more realistic appearance;
6. Automate the generation of the ellipsoids for a given model.

5. REFERENCES

[Angel00] Angel, E.: Interactive Computer Graphics: A Top-Down Approach with OpenGL, 2nd Edition, *Addison-Wesley*, 1997.

[Baraff98] Baraff, D., Witkin, A.: Large Steps in Cloth Simulation, *SIGGRAPH 98 Proceedings*, pp.43-54, 1998.

[Breen00] Breen, D.E.: A Survey of Cloth Modeling Methods, *Cloth Modeling and Animation*, *A. K. Peteres Ltd.*, Ch. 2, pp.19-53, 2000.

[Oshita01] Oshita, M., Makinouchi, A.: Real-Time Cloth Simulation with Sparse Particles, *SIGGRAPH 2001 Sketches and Applications*, pp.250, 2001.

[Perez99] Pérez-Urbiola, R., Rudomín, I.: Multi-Layer Implicit Garment Models, *Shape Modeling International Proceedings*, pp.66-71, 1999.

[Rudomin00] Rudomín, I., Melón, M.A.: Multi-Layer Garment Using Hybrid Models, *Visual 2000 Proceedings*, pp.118-128, 2000.

[Vassilev01] Vassilev, T., et al.: Efficient Cloth Model and Collisions Detection for Dressing Virtual People, *ACM/EG Games technology Conference*, 2001.

[Vlachos01] Vlachos, A., et al.: Curved PN triangles, *2001 ACM Symposium on Interactive 3D Graphics*, 2001.

[Volino00] Volino, P., Magnenat-Thalmann, N.: Virtual Clothing, *Springer Verlag*, 2000.

[Woo99] Woo, M., et al.: OpenGL Programming Guide, The Official Guide to Learning OpenGL, version 1.2, 3rd Edition, *Addison-Wesley*, 1999.