Distance fields applied to character animation

I. Rudomin and J. Castillo

Department of Computer Science, ITESM-CEM, Mexico

Abstract

A general hybrid geometric-physical method that is useful for animating characters with skin, clothing and a MPEG-4 compatible face is presented. This method uses an approximation to the underlying musculo-skeletal structure of the body and face to generate a distance field, used for collision detection purposes, and skin/clothing consisting of a particle-spring mesh. The results obtained deliver animation with plausible dynamics for fairly detailed models at around 60fps on a PIII computer with Nvidia Gforce2 graphics. The main contribution is to show that similar algorithms can be used for skin, clothing and facial animation, at interactive rates.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Animation

1. Introduction

Modelling and animating a complete character, including the character's skin, the clothing it is wearing, and its face, all at interactive rates is still a challenge for computer graphics. These problems can be characterised as instances of a more general problem: efficiently placing a particle-spring mesh, subject to physical forces, over an underlying unpenetrable structure.

One way of ensuring this is having the underlying structure generate a moving distance field, changing whenever the underlying structure moves, which we can evaluate to place each particle of skin or clothing. Since we want this to work interactively we need to simplify the representation of the underlying structure. Given that a few ellipsoids can approximate a body or skull, and that the distance of a particle to an ellipsoid can be calculated efficiently, we have chosen to implement the our system using ellipsoids.

The animation process we follow can be summarized as follows:

- The particles in the mesh are moved geometrically, (following the skeleton for the body or feature points for the face)
- The particle positions are adjusted to minimise mesh deformation, as well as other desired effects such as wind and gravity by the application of forces.
- Penetration of the underlying structure by any particle is corrected by using the distance field generated by the ellipsoids

In the following we will describe previous work and detail our proposal in regards to clothing and facial animation. Then we will report our current results, and finally we will have some conclussions.

2. Clothing

There are two major tasks that a clothed character simulation system must perform: model the behavior of cloth, and detect collisions between cloth and the character. In the following we discuss each of these problems in turn, as well as approaches that have been followed by other researchers to accelerate clothed character simulation. Then we will describe our approach.

2.1. Previous Work

The cloth simulation problem has been approached from different directions that can be classified as geometric, physically based, or a combination of both (hybrid)[1, 2]. The simplest physically-based methods use mass-spring particle systems to simulate the behavior of cloth. However, cloth is stiff, so solving the systems using explicit Euler or Runge-Kutta integration schemes require very small time-steps to be stable. One could also relax the stiffness of the springs, but this causes unnatural behavior. Baraff[3], handles this problem by using an implicit Euler integration scheme that allows the system to take large steps in the simulation, even for stiff springs as required to model the behavior of cloth.

However, each step involves significant computation, rendering the system more suitable (at present) for animation or off-line simulation than for interactive applications.

Different approaches to animating clothed characters in real time have been published: accelerating the simulation, using special purpose hardware features, and accelerating collision detection. Desbrun[4] uses a simplified version of Baraff's implicit integration method. Oshita[5] 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 vendor-specific hardware technique called PN-triangles is used. Vassilev[6] proposes 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.

2.2. Our approach: anatomical models for skin and clothing

It is becoming common to use anatomical models of humans and animals[7, 8]. In these methods, the skeleton and muscles of the animal or animal are approximated, sometimes using ellipsoids. These are usually not visible, but they are used as an underlying structure that can move the skin. In a similar manner, Rudomin[10] describes a method extending the above methods to allow the animation of a character wearing different layers of clothing. A group of implicit ellipsoids approximate the shape of a human character. See figure 1. Cloth

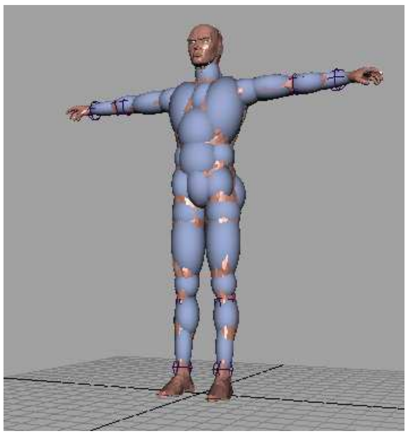

Figure 1: *Skin and ellipsoids approximating the character*

is modelled as a particle-spring mesh that can be animated geometrically and using physical simulation. The ellipsoids generate a scalar field that is used for colision detection.

The animation process is as follows:

- The particles in the cloth mesh are moved geometrically, (following the skeleton)
- The particle positions are adjusted to minimise mesh deformation, as well as other desired effects such as wind and gravity by the application of forces and using a solver (Euler, Runge Kutta).
- Penetration of the underlying structure by any particle is corrected by using an implicit function generated by the ellipsoids

In [11] some modifications to these methods that make them work at interactive rates are reported. The results are in the order of 60 fps, which, compared with the results previously obtained, of around 18 seconds per frame, indicated significant speedups. To achieve these results:

1. The distance to the parent ellipsoid is used to generate the scalar field instead of an expensive implicit function. This can be calculated efficiently with the appropriate data structure.
2. The simplest possible solver (explicit Euler) is used. This works if the cloth is not too stiff, and because the method is mostly geometrical, and physics is used only for minor corrections. Still, if more exact results are desirable, another solver could be used, even at the expense of speed.

See figure 2 for an example. Not all garments behave in the

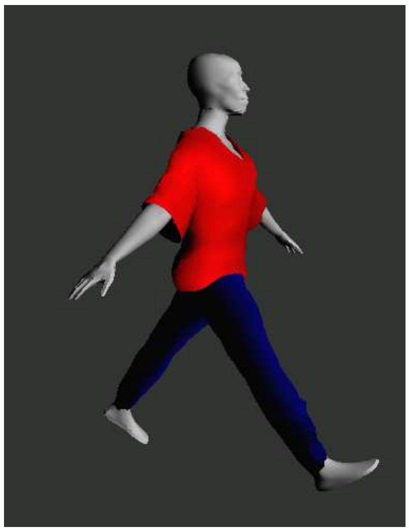

Figure 2: *Clothed character*

same way. We have provided for this variation with several slightly different versions of the clothing algorithm:

- We designed pants that stick to the body, but are not affected by wind or gravity: in this case adding spring forces (that keep the cloth together and disallow excessive stretching) as well as forces that keep cloth vertices and at a specific distance from the appropriate ellipsoid are necessary. Adding other forces, however, is unnecesary. See figures 2 and 3(a).

- The blouse we used is affected by wind and gravity: in this case these forces must be added, as well aas the spring forces that keep the cloth together, but forces that keep the cloth at a specific distance from the ellipsoid are not needed. See figures 2 and 3(b).
- We designed a skirt that has some vertices (the waist) that stick to the body, while the others are affected by wind and gravity: we specify for each vertex which of the previous two methods are used. If this were not done, the skirt might fall of the body. See figures 3(c) and (d).

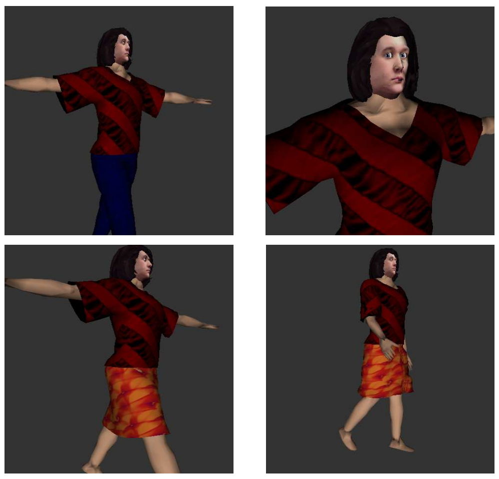
Figure 3: *The complete character: skin, clothing (pants or skirt, blouse) and head*

It is possible to apply this method not only to clothing, but also, with minor modifications, to the skin of the character. The skin mostly sticks to the ellipsoids (like the pants) except for features like hands and feet that are not changed in shape and are moved only geometrically. In these vertices we don't need to apply forces or check for collisions with the ellipsoids. See figure 4 for an illustration of the skinned character.

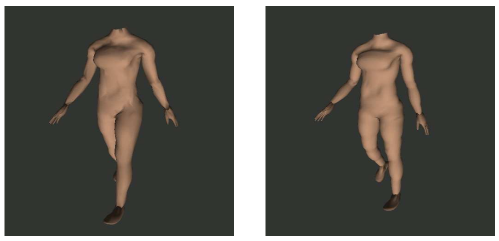
Figure 4: *The skin of the character, animated by the modified clothing algorithm*

3. Results

We have tested our method on a Pentium III/800M with 512M desktop computer with Gloria/DCC (PC1) and on a Pentium III/1.133G with 512M and Nvidia Gforce2Go/32MB notebook computer (PC2).

The results are presented in table 1 (the results listed are simulation time only, in seconds): From this table we can

Objects and #vertices	PC1	PC2
pants (464)	0.005	0.003
blouse(454)	0.004	0.003
both (918)	0.009	0.006
skirt (312)	0.004	0.002
body (1965)	0.016	0.012
torso(511)	0.005	0.004
arm(225)	0.001	0.002
leg(215)	0.002	0.002

Table 1: *Simulation Timing: in seconds*

conclude that simulation time allows us real time calculation, and, for a given processor-memory combination, depends on the complexity of the skin/garment as measured in the number of vertices. For the complete body, with skin, cloth and head, this results in frame-rates that are around 43 fps in PC1 and 58fps in PC2. This is consistent with and in fact exceeds our goal of obtaining simulation and rendering within interactive rates.

See figure 3 for an illustration of the complete clothed, skinned character, with a head.

4. Facial Animation

Several types of facial models have been developed, which can be roughly classified[12] either as parametric[13, 14], or muscle-based[15, 16, 17]. Parametric approaches represent and animate the facial model by directly modifying geometry using specific control parameters, while muscle-based approaches attempt to represent to some degree of accuracy the anatomic structure of the face and simulate the behavior of muscles to animate the model (within muscle-based approaches, non-anatomical models are usually called pseudo-muscle-based).

Many of the approaches, in all of the above mentioned categories, essentially consider the face as a flexible mask that can be interpolated or pulled by the muscles without any constraints. Anatomy, however, just like for bodies, places some important restrictions on facial movement, since, underneath the skin, there are rigid structures (skull and jaw) that should not be penetrated by this mask. Models that do not take this underlying structure into account are limited in their ability to synthesize realistic facial dynamics. Some models of the face are anatomical in the sense used above for animals and humans[7, 8, 9]. One example is Kähler[18], who uses a muscle model that incorporates different types of muscles and the effects of bulging and intertwining muscle fibers. The influence of muscle contraction onto the skin is simulated using a mass-spring system that connects the skull, muscle, and skin layers.

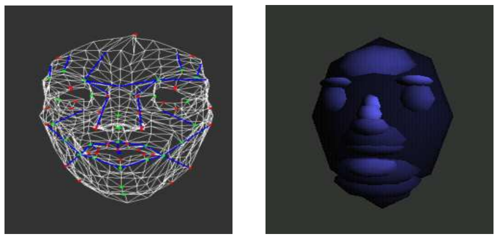
Figure 5: *(a) Wireframe of face, muscles and MPEG-4 control points; (b) Associated ellipsoids.*

4.1. Our approach

In a face, not all movement is directly due to the movement of the skeleton, but rather, to the movement of facial muscles. To be precise, in reality, the body is also moved by muscles, not by the skeleton; however it is standard practice in graphics to move the skeleton. While this works well for body animation (although it does not model all body movement) it does not work for faces.

Our facial animation player was built for a “Speech Driven Facial Animation” project (see acknowledments at the end). This project involves building an integral system capable of generating animations with perceptually realistic dynamics, including the individualized nuances, of three-dimensional human faces driven by speech. The system is able to capture very short phenomena in the orofacial dynamics of a given speaker by tracking the 3-D location of various MPEG-4[20] facial points through stereovision. Using the captured speech and video database, new speech is used to predict 3-D facial dynamics by means of an attractive nearest-neighbor algorithm. A statistical analysis of the facial movement data is performed to reduce dimensionality and for filtering.

The data is generated as a series of 3D positions for selected (27) MPEG-4 feature points, every $1/60^{th}$ of a second. Using the MPEG-4 facial Action Parameters (FAPs) as is common in many systems would lose the subtle cues for coarticulation present in the data, so it was not an option for us. We chose to use MPEG-4 feature points and use them to animate the face in a manner similar to Kshirsagar[19]. In our case, muscle and jaw animation, and in fact the animation of the whole facial mesh must be inferred from the feature points. In figure 5 we can see the MPEG-4 feature points, (in the color version of this figure one can see in red the feature points that are used to determine facial proportions, in green the ones used to track or predict movement) overlayed on the wireframe of the face model. The associated “muscles” (in blue in the color version) are also shown in this figure5(a). The ellipsoidal structure for the face is shown in figure 5(b). Our approach is a pseudo-muscle approach that attempts to have some extra anatomical basis (solid skull-jaw). The “muscles” are realy additional springs that can be contracted or elongated as in standard pseudo-muscle approaches[12], but that in fact we drive directly from the MPEG-4 feature points. We do not perform an anatomically realistic geometric and physical modelling of the muscles (like Kähler) although this would be possible using the ellipsoids in the system. We use these ellipsoids instead to represent an approximate geometry of the skull and the jaw to control collisions. This is a far simpler (and presumably faster, which was our main constraint). If the constraints for non-penetration of skull and jaw by the vertices in the facial model are not taken into account, the facial dynamics will be wrong (in particular in parts of the face that are not feature points).

Implementation involved using a basic model by Parke and Waters[12] and modifying it into a generic facial model where we specified 64 MPEG-4 feature points. The model contains a mesh of 876 triangles and 28 muscles to allow facial expressions and movement.

This generic model must be adjusted to the proportions recorded in the animation data, and we did this by interpolating between the 3D coordinates of MPEG-4 feature points of the generic neutral face and the ones in the beginning of the animation data, (which correspond to the subject’s neutral face). At each step, these interpolated values are in turn used to smoothly pull all the vertices of the generic mesh towards the individualized model by treating the mesh as a particle-spring system and using a PDE solver such as Euler or Runge-Kutta at each step until the system is stabilized. At the end of this process we have obtained an individualized model. We apply the photograph of the subject as a texture. This is done interactively using a custom program. The resulting textured animatable facial model and can be seen in figure 6.

For animation, applying the feature point information directly to the facial model is not sufficient since all the remaining vertices in the facial mesh must also be moved smoothly to produce an overall natural movement. We could follow a procedure similar to the one used to fit the model, but it turns out it is too slow. What we do instead, is associate each of the 28 muscles of the generic face with one or more MPEG-4 feature points (as a head or a tail), a predetermined

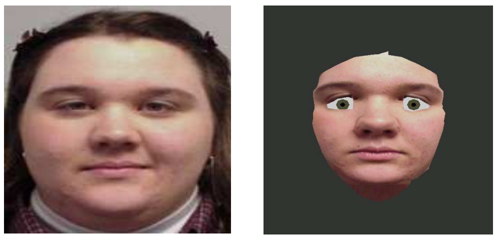

Figure 6: *(a) Texture used (b) Textured facial model*

area (a list of vertices to modify) and a stiffness constant. This is similar to what is done by Kshirsagar [19]. The list of vertices associated to each muscle is determined once (when the generic face model is modified to generate a new model for a particular speaker) and subsequently stored in a data structure.

The muscles are added to the model as additional springs and are processed in the same manner as the rest of the springs in the mesh.

To improve the realism of the animation, the player is augmented with the underlying ellipsoidal structure described previously. What we did was to take the cloth-skin algorithm described in the previous section and adapt it for facial animation. When applied to facial animation, this ellipsoid representation avoids unnatural penetration of the underlying skull-jaw structure that is often present in other approaches. As a result, this player is able to model facial dynamics more accurately.

Before we apply our algorithm we must:

- Assign each particle in the face-mesh to the appropriate feature point. The influence of neighboring feature points on other vertices is determined during start-up. This influence is weighted based on the distance of every vertex to those feature points within a certain distance. As a result, a vertex may be influenced by more than one feature point. An exception is made with the vertices that define the lips, since they closely follow specific feature points. This is done manually, but it must be done only once.
- Approximate the skull and the jaw using ellipsoids, and assign vertices to the appropriate ellipsoids as we did for the body. This again was done manually, just as for the body.

Our algorithm takes the following steps, every $1/60^{th}$ of a second:

1. Modify the particle positions according to the changes in the feature point positions supplied in the data.
2. Jaw rotation is determined by calculating the angular difference between the current position of the feature points located below the nose and on the chin, and their corresponding positions during start-up. When the ellipsoids that define the jaw rotate, all the vertices associated to them that were not modified in the previous step are transformed accordingly.
3. To incorporate the effect of spring forces, springs are defined for every edge in the face mesh and for every muscle. Forces are calculated for these springs and the resulting displacements are applied to the vertices.
4. Use the distance field generated by the ellipsoids to avoid ellipsoid penetration. If a vertex is determined to penetrate an ellipsoid, the vertex is moved to the ellipsoid's surface.

5. Results

We have tested our method on a Pentium III/800M with 512M desktop computer with Gloria/DCC (PC1) and on a Pentium III/1.133G with 512M and Nvidia Gforce2Go/32MB notebook computer (PC2).

The results are presented in table 2 (the results listed are simulation time only, in seconds):

Objects and #vertices	PC1	PC2
mask (526, no speech)	0.003	0.002
mask (526, with speech)	0.005	0.004

Table 2: *Simulation Timing for Facial Animation (in seconds)*

For facial animation, this results in frame-rates that are fixed at 60fps so that synchronization with sound is achieved in both PC1 and PC2. This is consistent with and in fact exceeds our goal of obtaining simulation and rendering within interactive rates. See figure 7 for several frames of the animation. Note that hair (not animated), teeth, tongue have been added. Also, eyes are now textured and blink.

6. Conclusions

The results show that this hybrid geometrical-physical method using distance fields generated from an ellipsoidal approximation to the underlying skeletal-muscular structure is fast, stable and portable. It works well for skinned and clothed characters with facial animation, for real time applications, where interaction is more important than accuracy.

We are working on:

- Adding level of detail to the spring-mesh by using subdivision algorithms.
- Adapting the subdivision algorithms to add wrinkles.
- Using other solvers (besides Euler) to obtain physically-correct results without reducing the frame rates.
- Exploring the use of adaptive distance fields.
- Applying a similar method to hair.

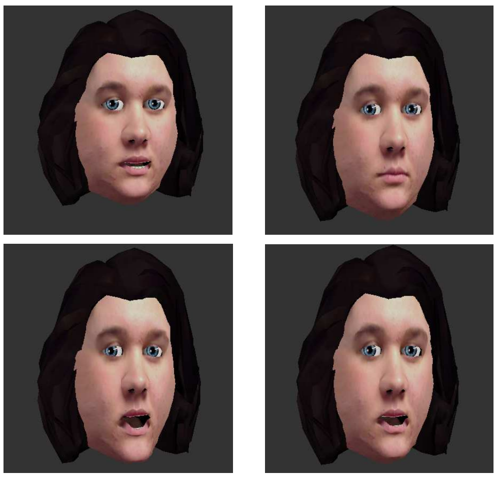

Figure 7: *The animated facial model*

7. Acknowledgements

I would like to thank Nadia Magnenat-Thalmann, Daniel Thalmann and Pascal Volino for the useful discussions and advice regarding the work presented in this paper, and Oscar García, Ricardo Gutierrez-Ozuna and the whole team at Wright State for their facial animation data.

This work was partly supported by NSF-CONACyT project C30033, “Speech Driven Facial Animation”.

References

1. Volino P. and Magnenat-Thalmann, N.: Virtual Clothing. Springer Verlag, 2000
2. House D. and Breen D., editors: Cloth Modeling and Animation. A.K. Peters, 2000
3. Baraff, D. and Witkin, A.: Large Steps in Cloth Simulation. Computer Graphics, SIGGRAPH ’98 Proceedings, pp.43
4. Desbrun, M, Scröder, P. and Barr, A.: Interactive Animation of Structured Deformable Objects. Graphics Interface ’99 Proceedings, pp.1
5. Oshita, M., Makinouchi, A.: Real-Time Cloth Simulation with Sparse Particles, SIGGRAPH 2001 Sketches and Applications, pp.250, 2001.
6. Vassilev, T., Spanlag, B., Chrisanthou, Y.: Efficient Cloth Model and Collisions Detection for Dressing Virtual People, ACM/EG Games technology Conference, 2001.
7. Wilhelms, J. and Van Gelder, A.: Anatomically Based Modeling, Computer Graphics (SIGGRAPH ’97 Proceedings), pp. 173-180, 1997.
8. Scheepers, F. Parent, R., Carlson, W. and May, S.: Anatomy Based Modeling of the Human Musculature, Computer Graphics (SIGGRAPH ’97 Proceedings), pp. 163-172, 1997.
9. Thalmann, D., Shen, J. and Chauvineau, E.: Fast Realistic Human Body Deformations for Animation and VR Applications, in Computer Graphics International proceedings, 1996 Pohang, Korea, pp. 166-176
10. Rudomin, I., Pérez-Urbiola, R., Melón, M., and Castillo, J.: Multilayer garments using isosurfaces and physics, The Journal of Visualization and Computer Animation Volume 12, Issue 4, 2001. (Special Issue: The best papers of Visual 2000. Issue Edited by Isaac Rudomin.)
11. Rudomin, I. and Castillo, J.: Realtime clothing: geometry and physics, in WSCG 2002 Posters, ISBN 80-903100-0-1,WSCG 2002, Plzen, pp 45-48
12. Parke, F. and Waters, K.:Computer Facial Animation, A. K. Peters, Wellesley, Massachusetts, 1996.
13. Parke, F.:Parameterized models for facial animation, in IEEE Computer Graphics and Applications, 1982, 2(9), pp. 61-68.
14. Parke, F.:Computer generated animation of faces, MS Thesis Technical Report, UTEC-CSC-72-120, Department of Computer Science, University of Utah, Salt Lake City, Utah.
15. Waters, K. and Terzopoulos, D.:A Physical Model of Facial Tissue and Muscle Articulation, in proceedings of the First Conference of Visualization in Biomedical Computing, Atlanta, Georgia, 1990, pp. 77-82.
16. Waters, K.:A muscle model for animating three-dimensional facial expression, in Computer Graphics (SIGGRAPH ’97 Proceedings), pp. 17-24.
17. Nedel, L. and Thalmann, D.:Real Time Muscle Deformations Using Mass-Spring Systems, in proccedings CGI’98, IEEE Computer Society Press.
18. Kähler K., Haber, J. and Seidel, H.:Geometry-based Muscle Modeling for Facial Animation, in proceedings Graphics Interface 2001, pp. 27-36
19. Kshirsagar, S., Garchery, S. Magnenat-Thalmann, N.:Feature point based mesh deformation applied to MPEG-4 facial animation, in Deformable Avatars: IFIP TC5/Wg5.10 Deform’2000 Workshop, Geneva, Switzerland
20. ISO, Overview of the MPEG-4 Standard, at http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm 1 1 1 2 2 2 2, 4 2, 4 4 2 2 3, 4 3 3 3 3 3 4 4, 5 4