Fast Realistic Human Body Deformations for Animation and VR Applications*

Daniel Thalmann, Jianhua Shen, Eric Chauvineau

Computer Graphics Lab (LIG), Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland

Virtual Actors now play an important role in Computer-generated films, Virtual Environments, Telecooperative work, and multimedia. In order to make these actors realistic, it is essential to represent their body shape during the motion. In this paper, we present different methods for representing realistic deformations for virtual humans with various characteristics: sex, age, height, weight. Our methods based on a combination of metaballs and splines could be applied to frame-by-frame computer generated-films and Virtual Environments. Several examples will be presented: autonomous actors, animation based on Flock of birds, networked Virtual Environments.

1 Human Body Built from Implicit Primitives

Virtual Actors now play an important role in computer-generated films, virtual environments, telecooperative work, and multimedia. This research area still remains one of the most difficult and challenging problems, since a human being is a very complex object and our eyes are especially sensitive to the human figure.

There are two major difficulties in rendering human animation: motion of the hierarchical structure or skeleton and deformation of the body. This paper addresses the second problem. Modeling and deformation of 3D characters and especially human bodies during the animation process is an important but difficult problem. Researchers have devoted significant efforts to the representation and deformation of the human body shape. Broadly, we can classify their models into two categories: the surface model and the multi-layered model.

The surface model [1] is conceptually simple, containing a skeleton and outer skin layer. The envelope is composed of planar or curved patches. One problem is that this model requires the tedious input of the significant points or vertices that define the surface. With the advent of laser scanning systems, meshes of extreme complexity are rapidly becoming common place. However, such meshes are notoriously expensive to store, transmit, render, and are awkward to animate. It is very difficult to modify these meshes to a different shape. Needless to say, digitizing facilities are expensive and it is not uncommon that there is no access to a scanning device when 3-D body shape data are required. Another main problem is that it is hard to control the realistic evolution of the surface across joints. Surface singularities or anomalies can easily be produced. Simple observation of human skin in motion reveals that the deformation of the outer skin envelope results from many other factors besides the skeleton configuration.

The Multi-layered model [2] contains a skeleton layer, intermediate layers which simulate the physical behavior of muscle, bone, fat tissue, etc., and a skin layer. Since the overall appearance of a human body is very much influenced by its internal muscle structures, the layered model is the most promising for realistic human animation. The key advantage of the layered methodology is that once the layered character is constructed, only the underlying skeleton need be scripted for an animation; consistent yet expressive shape deformations are generated automatically.

Implicit surface techniques offer the opportunity of modeling and animating complex organic shapes at a fraction of the cost in data points of more common patching techniques. Implicit surface are typically defined by starting with simple building block functions (called *primitive*) and by creating new implicit functions using the sum, min, or max of simpler functions. The final object is constructed by blending the primitives, and as the primitives are moved and deformed the resulting blended surface changes shape. A particular subset of implicit surfaces, called *soft objects* [3] (or *metaballs*[4], *blobs* [5], *convolution surfaces* [6]), have received increasing attention in computer graphics.

We have developed an interactive system, named "**Body Builder**", for interactive design human bodies from scratch or by modifying existing models. Ellipsoidal metaballs and ellipsoids are employed to

*Published in Proc. Computer Graphics International IEEE Computer Society Press, June 1996, pp. 166-174

represent the gross shape of bone, muscle and fat tissue. Each primitive is attached to its proximal joint, defined in the joint local coordinate system of the underlying human skeleton.

The implicit surface technique is inherent to interactive design. The designer starts with a rough shape consisting of a few primitives, then he/she adds details by simply editing primitives: add, delete, transform, adjust the parameters of primitives. Because of their nature, we found that implicit primitives can build human prototypes very efficiently. Since files for primitive models are typically at least two to three orders of magnitude smaller than those modeled with polygon or parametric patches, a compact, parameterized body model suitable for communication, can be obtained. Figure 1 shows an example.

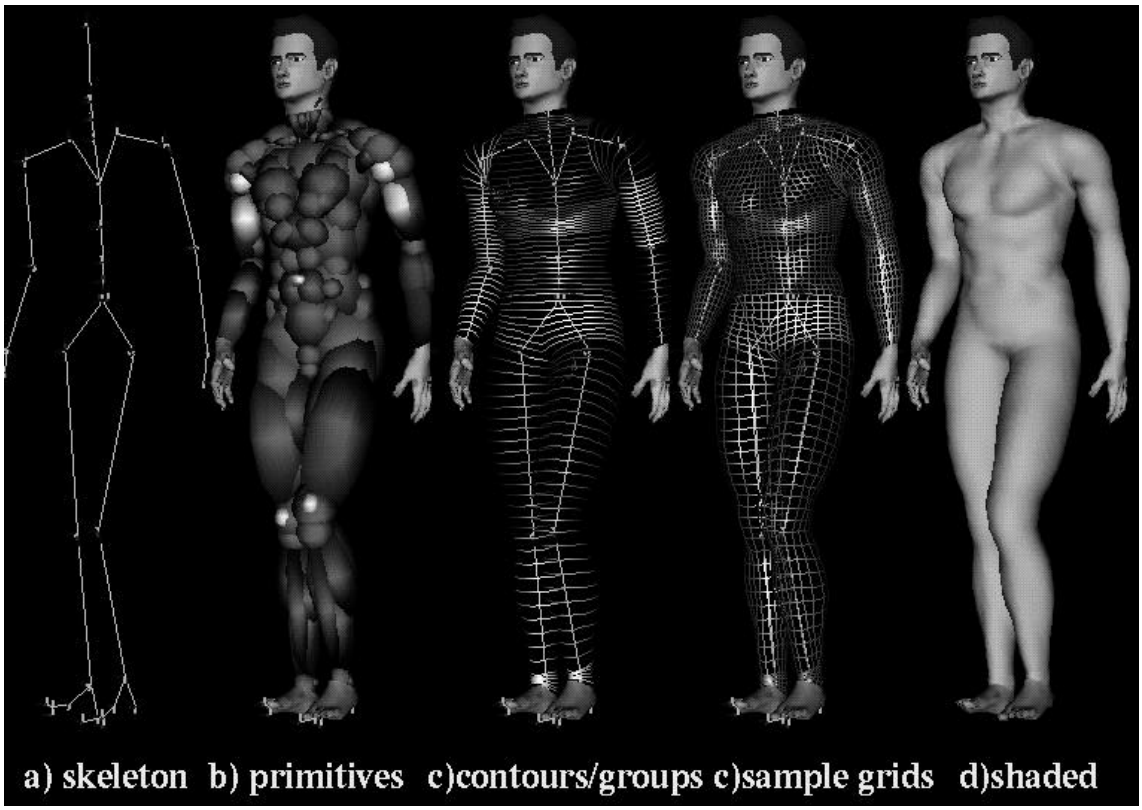

Fig. 1. Layered construction of human body

2 Contours Extraction from Implicit Primitives

The ensemble of volume primitives implicitly defines an object's shape. There is always a prevalent need for an explicit representation. Sampling and visualizing an implicit surface has attracted much interest recently in the graphics community, including particle-based sampling techniques[7] and polygonalization techniques[3,8].

Our approach [9] profits from the fixed topology of the human skeleton. Human limbs exhibit a cylindrical topology and the underlying skeleton provides a natural centric axis upon which a number of cross-sections can be defined. Each limb link is associated with a number of contours. We can automatically extract cross-sectional skin contours using the ray casting method, as illustrated in Figure 2.

We cast rays in a star-shaped manner for one contour, with ray origins sitting on the skeleton link. For each ray, we compute the outermost intersection point with the implicit surface surrounding the link. The intersection is a sample point on the cross-section contour.

We first systematically configure the ray origin and direction for every contour point to be sampled. Two rules are used to configure cross-sectional contours: *joint angle driven configuration* and *boundary driven configuration*. The first rule is applied between two links of a skin piece; the second is only applied to a few boundary contours of a skin piece to ensure a natural transition across common boundaries of adjacent skin pieces (Figure 3).

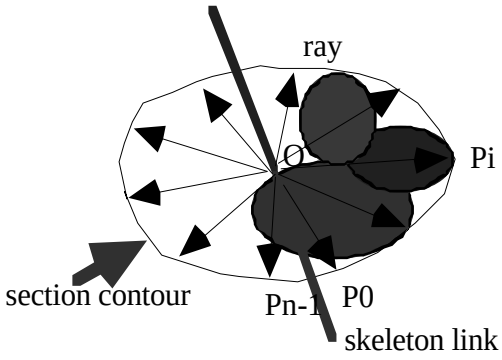

Fig. 2. Sampling a contour by ray-casting (shaded ellipse represent volume primitives)

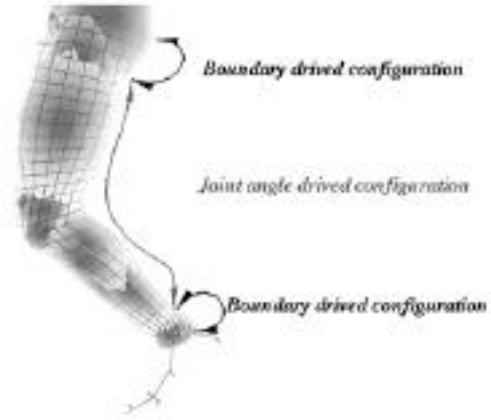

Fig. 3. Cross-sectional configuration of the leg

2.1 Joint angle driven configuration

This rule uses the angle between two connected links to drive the position and orientation of the cross-section planes in-between. In Fig. 4a, $\mathbf{L}_1$ is the direction of the upper link, $\mathbf{L}_2$ the lower. $\mathbf{N}_u$ and $\mathbf{N}_l$ are two normals of the cross-section planes at the link ends. We set the orientation of the cross-section plane that passes through the joint to be the bisection plane of the two links. Let this bisection plane normal be $\mathbf{N}_0$. Suppose O_i and $\mathbf{N}_i$ are the center and normal respectively of the *i*-th cross-section plane along the upper link. O_i is distributed evenly along the upper link by linear distance interpolation of the upper link length, and $\mathbf{N}_i$ by direction interpolation of two end normals $\mathbf{N}_0$, $\mathbf{N}_u$. For each point on the *i*-th contour to be sampled, we cast a ray with origin O_i, and circulate direction around $\mathbf{N}_i$ (Figure 4b).

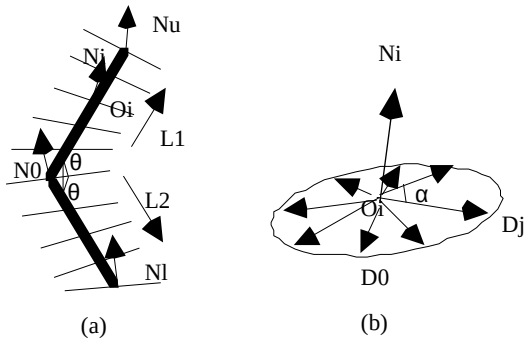

Fig. 4. Cross-sectional plane orientation and ray distribution

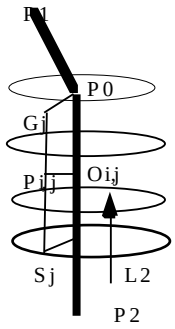

Fig. 5. Calculate ray origins & directions of intermediate contours

In the case of $\mathbf{L}_1 \| \mathbf{L}_2$, simply set $\mathbf{N}_0 = \mathbf{L}_1$. If the upper link is an end link in the skeleton, set $\mathbf{N}_u = \mathbf{L}_1$. Otherwise, set $\mathbf{N}_u$ the bisection plane normal of that joint, same as $\mathbf{N}_0$. The lower link is treated in the same way. This process is applied repeatedly to all links in the skeleton.

2.2 Boundary driven configuration

The *joint angle driven configuration* method regularly orientates the cross-sections of a skin piece in space according to the current joint angles of the related links, but it does not take into account the shape of adjacent skin piece's boundary. The idea behind the *boundary driven configuration* is to mimic a "*force field*" that, by creating a few intermediate contours inbetween a *start contour* and a *goal contour*, make the *start contour* tend to the *goal contour*. This ensures a natural, smooth shape transition across common boundaries. Here the *start contour* and *goal contour* belong to the boundary of two adjacent skin pieces respectively.

Let $\mathbf{G} = \{G_1, G_2, \cdots, G_{n1}\}$ be a *goal contour* of skin piece A surrounding link $\boldsymbol{P_0P_1}$, and $\mathbf{S} = \{S_1, S_2, \cdots, S_{n2}\}$ be a *start contour* of adjacent skin piece B surrounding link $\boldsymbol{P_0P_2}$ (Fig.). Suppose we want to generate k intermediate contours between $\mathbf{S}$ and $\mathbf{G}$. If $n_1 \neq n_2$, we can resample n_2 points along the segments of the *goal contour* by parameterizing the perimeter of $\mathbf{G}$ and locating new sample points along the contour in proportion to the normalized length. Now assume $n_1 = n_2 = n$, and preprocessing of the boundary match has been done, so that $\mathbf{S}$ and $\mathbf{G}$ run in the same direction and the span distance $< G_1, S_1 > = \min_{j=1}^{n} \| < G_1, S_j > \|$.

Let $\mathbf{C_i} = \{C_{i1}, C_{i2}, \cdots, C_{in}\}$ be the i-th intermediate contour to be generated. In order to sample $C_{ij}\ (j = 1 \cdots n)$, we calculate the ray origin O_{ij} and direction $\mathbf{D}_{ij}$ as follows: let $P_{i,j} = S_j + \frac{i}{k+1}(G_j - S_j)$, O_{ij} is obtained by projecting point $P_{i,j}$ orthogonally to link $\boldsymbol{P_0P_2}$, $\mathbf{D}_{ij} = O_{ij} - P_{ij}$. Since we use distance interpolation between $\mathbf{S}$ and $\mathbf{G}$, the intermediate contours sampled are evenly distributed in space, and form a natural tendency from *start contour* to *goal contour*.

3 Realistic Body Deformation by Frame-Based Resampling

3.1 Grouping primitives and contours

When we move two blendable primitives near each other, they blend together into one smooth shape. Sometimes, however, we want some primitives to blend together while others stay apart. A classic case would be the legs of a character. Each leg should not blend with the other leg. One solution is to assign each primitive to a group. Beier[BEIE93] have discussed this technique of avoiding unwanted blending. Each group will blend with primitives in its own group or in the root group. We define fifteen groups covering whole body for fine control of blending: *upper_torso, lower_torso, hip, left_shoulder, right_shoulder, left_upper_arm, left_lower_arm, right_upper_arm, right_lower_arm, left_upper_leg, left_lower_leg, right_upper_leg, right_lower_leg, left_ankle, right_ankle*. Each group associates a bunch of contours that represent its specific skin region. Figure 1c shows the correspondence. Contours belonging to the same group are displayed with the same color. Each primitive is assigned to a group according to its attachment to the skeleton. Some primitives, located near a joint, can fall into several groups simultaneously, as they may have contributions to multiple groups. To sample a contour, we only need to consider the primitives in its associated group. By mapping groups to contours, we can successfully avoid unwanted blending (Figure 6).

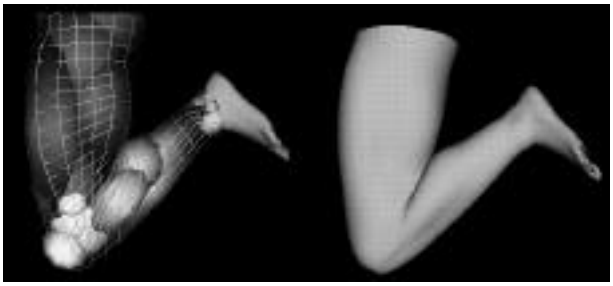

Fig. 6. Grouping primitives and contours to avoid unwanted blending
(group color of primitives: pink: *upper-leg,* green: *lower leg*, yellow: both)

3.2 Deform primitives

In our model, we introduce the concept of *deformable primitive,* whose parameters are a function of the underlying skeleton state, to mimic the contraction and release behavior of muscles. Each deformable primitive is associated with *a reference joint,* whose value dynamically determines the center, orientation and shape of that primitive. Usually we can easily map the association based on anatomic structures. For example, the reference joint of a primitive representing the biceps muscle should be *elbow_flexion.*

The state of an ellipsoidal primitive ***E*** at any time is specified by a set of nine parameters, $G = \{a, b, c, O_x, O_y, O_z, \phi_x, \phi_y, \phi_z\}$ (we keep the weight constant during animation). The limit of its reference joint ***J*** value is $[\alpha_{\min}, \alpha_{\max}]$. We define $\alpha_0\ (\alpha_{\min} \leq \alpha_0 \leq \alpha_{\max})$ as the value corresponding to

the relaxed state of ***J*** (i.e., the skeleton in default posture). For most joints, $\alpha_0 = 0$. Let α be the current joint value of ***J***. In our skeleton, when $\alpha \in [\alpha_{min}, \alpha_0]$, it corresponds to the forward or upward movements of ***J***; $\alpha \in [\alpha_0, \alpha_{max}]$ corresponds to the backward or downward movements. Deformable primitives change their states as follows.

Let G_0, G_{min}, G_{max} be the predefined state of ***E*** when $\alpha = \alpha_{min}, \alpha_0, \alpha_{max}$ respectively.

If $\alpha \in [\alpha_0, \alpha_{max}]$, then $t = \dfrac{\alpha - \alpha_0}{\alpha_{max} - \alpha_0}$, $G = G_0 + f(t)(G_{max} - G_0)$;

if $\alpha \in [\alpha_{min}, \alpha_0]$, then $t = \dfrac{\alpha - \alpha_{min}}{\alpha_0 - \alpha_{min}}$, and $G = G_0 + f(t)(G_{min} - G_0)$.

$f(t) \in [0,1]$ is a continuous function. Currently a user can choose one of three types of functions: linear $f(t) = t$, parabolic $f(t) = t^2$ and hyperbolic $f(t) = \sqrt{t}$. Here the state of ***E*** is interpolated between the default and extreme state of its reference joint. We distinguish two phases of α because the muscle deformation may have different behaviors in different directions.

During animation, when the underlying skeleton moves, all primitives attached to their relevant joints undergo the joint hierarchy transformations as rigid body motions. An implicitly defined surface by volume primitives is sampled using ray-casting on semi-regular cylindrical grids at each frame. These sample points are used directly as cubic B-spline control points to smooth out the skin surface. Individual B-spline patches are triangulated, and these triangular meshes are stitched together to connect different parts of the human body for final rendering and output.

4 Realtime Deformation by Manipulating Contours

Considering VR applications, where frame rate should be between 15 and 25 frames per second, we simplify our previous method to obtain realtime deformations for the complete body.

4.1 New approach

This approach is based on contours deformation. First we compute these contours points in a given position of the body (we call it 'initial position'). At this point, we don't compute anymore B-spline patches and triangulate these surfaces: contours points are used directly for body triangles mesh. As we already explained, each contour belongs to one body part. For convenient reasons, we still consider these different body parts and use them to construct the final 3D surface.

Figure 7 shows the new triangles mesh made by contours computed in 'initial position'. Hands, feet and head are not considered in our method : they are represented by undeformable triangles mesh.

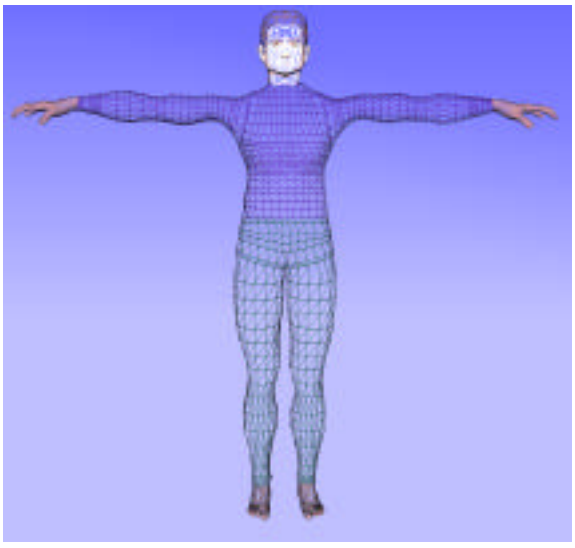

Fig. 7. Body mesh in initial position

Then, during realtime animation, we deform directly contours still using joint angle driven configuration and boundary driven configuration methods. By this way, we avoid contours computation from implicit surfaces, which is, with B-spline patches triangulation, the most time-consuming operation.

4.2 Implementation

In order to have an efficient implementation, we use the Performer library. This realtime graphics toolkit enables application to achieve maximum graphics performance from all Silicon Graphics workstations. It is easy with this library to create a 3D scene with many objects coming from different modelers. But, except morphing, no deformation capabilities have been proposed by Performer.

Consequently, we define our own structure compatible with Performer data arrangement (linear). By using index array to define triangle meshes, we avoid points duplication and make efficient memory management. Figure 8 shows our implementation of deformable body using Performer library.

It is also easy with Performer environment to apply texture on a 3D surface. By using our model configuration, we can apply different textures on each body part. Using hardware texture mapping, performance is the same than without texture, but enable texturing is a good way to increase surface realism. Figure 9 is almost the same as Figure 8; the only difference is, this time, texture mapping on the body envelope.

Fig. 8. Body deformation based on Performer library

Fig. 9. Body surface with texturing

4.3 Level-of-detail management

In realtime applications, an efficient way to reduce computing time is to manage different levels-of-detail (LOD) for one object: considering the distance between the camera and our object, we display this object in the best resolution for this distance.

As we know, the body envelope is one unique surface composed of eleven triangles meshes. Dimensions of each triangle mesh is given by the number of contours and the number of points per contour. By modify these numbers, it is possible to get different LODs for each body part and thus for all the body. Figure 10 shows three different LODs of the body surface which can be used in realtime applications using Performer toolkit.

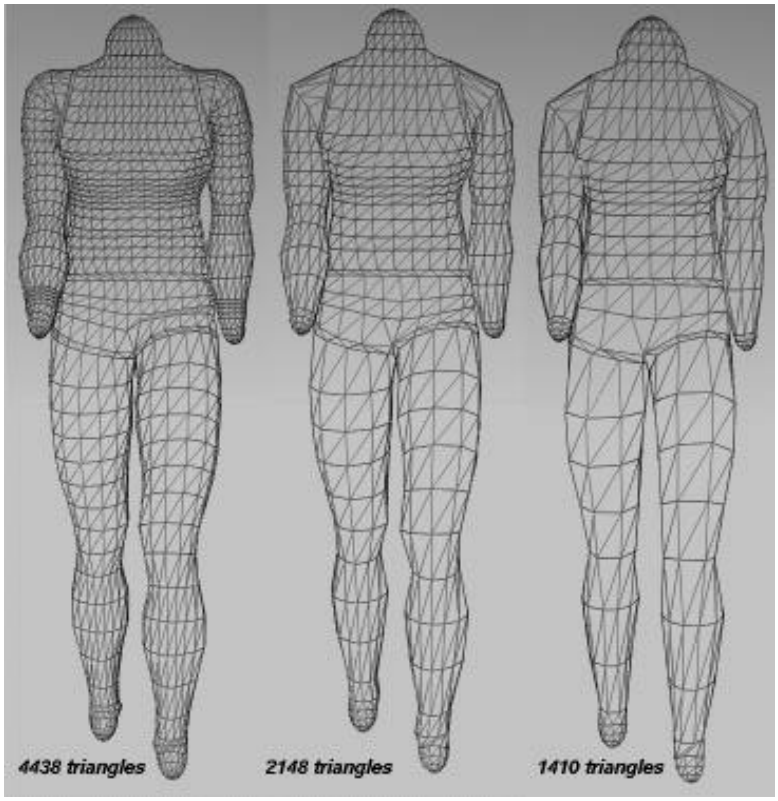

Fig. 10. Different LODs for body surface

4.4 Applications

Realtime deformable actors are a key issue for many multimedia applications: virtual actors for digital television [10], games [11] and VR applications. In order to provide maximum immersion for using virtual actors, it is important to use the most realistic human representation possible with real time considerations. Therefore, it is preferred to use the deformed body surfaces as envelops attached to the human skeleton, rather than simple representations such as basic geometric primitives. It is also necessary to simulate the articulations of the body, i.e. the joints to connect the body limbs with realistic constraints. In particular, we use them extensively in the VLNET system. The VLNET system [12] supports a networked shared virtual environment that allows multiple users to interact with each other and their surrounding in real time. The users are represented by 3D virtual human actors, with realistic appearances and articulations. The actors have similar appearance and behaviors with the real humans, to support the sense of presence of the users in the environment. In the environment, each participant is represented by a 3D virtual body that resembles the user. Each user sees the virtual environment through the eyes of her body. The user controls the movement of her virtual actor by various input devices, such as mouse, SpaceBall. Stereo display devices such as shutter glasses and head-mounted display can also be used for more realistic feeling in the environment. We also include additional virtual autonomous actors in the environment. The autonomous actors are connected to the system in the same way as human participants, and enhance the usability of the environment by providing services such as replacing missing partners, providing services such as helping in navigation. As these virtual actors are not guided by the users, they should have sufficient behaviors to act autonomously to accomplish their tasks. This requires building behaviors for motion, as well as appropriate mechanisms for interaction.

5 Automate Shape Variations by Scaling Operators

In order to cultivate a more lively awareness of human variety, it would be interesting and useful that by feeding some parameters into our model, the model can approximately reveal quantitative variation of human bodies.

We introduce scaling operators to automatically generate human bodies with different sizes and proportions. Because in our model the shape of human body is specified by the parameters of primitives,

body scaling is straightforward. Two procedures are required for scaling the human body: first, five normalized parameters are defined to scale the skeleton template to accommodate variations in age, sex and race (Figure 11); second, the shape parameters of all primitives are scaled accordingly.
Let the skeleton be in a default posture and assume **x, y, z** axes of the global frame represent *lateral*, *frontal* and *high* direction of the skeleton respectively. Five parameters are used in a two-step process to scale the skeleton:

A) a uniform scaling is made along x, y, and z with the `total_height`.

B) specific scaling are made along the following independent directions:

- frontal: along y axis with `frontal_scaling`
- lateral: `high_lateral_scaling` and `low_lateral_scaling` are proposed to differentiate the higher body from the lower body (useful for gender characterization)
- vertical: while keeping the same total height, the skeleton can be characterized with the parameter `spine_origin_ratio` which expresses the ratio of spine origin height over the total height

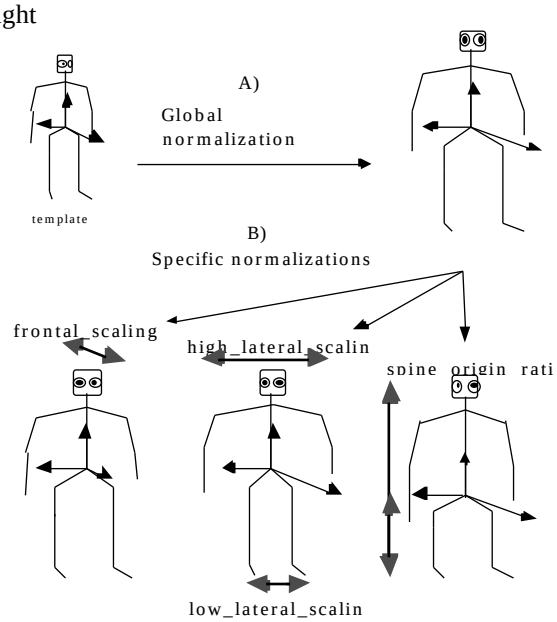

Fig. 11. Skeleton scaling

We associate a *tag* to indicate the correspondence of each principal direction of a primitive to the frontal, lateral, or vertical direction of the skeleton, and then selectively scale the corresponding axis length and center according to the current skeleton ratio while keeping the orientation. The tag of an ellipsoidal primitive is established by finding the dominant component of its principle directions in the global frame. That is, if $\mathbf{R}$ represents one principle direction in global frame, then

$$\mathbf{R} \Leftrightarrow \textit{lateral} \quad \textit{if } |\mathbf{R}_x| = \max\left(|\mathbf{R}_x|, |\mathbf{R}_y|, |\mathbf{R}_z|\right),$$

$$\mathbf{R} \Leftrightarrow \textit{frontal} \quad \textit{if } |\mathbf{R}_y| = \max\left(|\mathbf{R}_x|, |\mathbf{R}_y|, |\mathbf{R}_z|\right),$$

$$\mathbf{R} \Leftrightarrow \textit{high} \quad \textit{if } |\mathbf{R}_z| = \max\left(|\mathbf{R}_x|, |\mathbf{R}_y|, |\mathbf{R}_z|\right)$$

Users can either type in the exact new dimension of the skeleton, then scaling operation will automatically generate a scaled body model. They may also interactively apply these five scaling operations successively to get a satisfactory new model. The result model can be a new prototype for further editing.

6 Conclusion

In this paper, we have proposed a method for representing human bodies for any values of the joints angles. Not only the method provides realistic results, but the performance is very good: about 5 seconds a frame for one actor. We have also described a simplification of the method for realtime animation. This method is currently used for several applications in behavioral animation and VR. Our VLNET system for networked multimedia environment and telecooperative work also takes advantages of the method.

References

[1] Magnenat-Thalmann, N., Thalmann, D. "The Direction of Synthetic Actors in the Film Rendez-vous à Montréal ", *IEEE Computer Graphics and Applications*, Vol.7, No 12, 1987, pp.9-19.

[2] Chadwick, J.E., Haumann, D.R., and Parent, R.E. "Layered construction for deformable animated characte r ", *Computer Graphics*, 23, (3), 1989, pp.243-252

[3] Wyvill, G, McPheeters, C. and Wyvill, B. "Data Structure for soft objects", *The Visual Computer*, Vol.2, No.4,1986.

[4] Nishimura H., Hirai M., Kawai T., et al ."Object modeling by distribution function and a method of image generation", *Trans. IECE*, J68-D(4), 1985, pp.718- 725.

[5] Blinn, J.F. "A generalization of Algebraic Surface Drawing", *ACM Transactions on Graphics*. 1,3, 1982, 235-256

[6] Bloomenthal J., Shoemaker, K.,"Covolution Surfaces", *Computer Graphics*, Vol 25, no.4, 1991, pp.251-256.

[7] Witkin, A.P. and Heckbert, P.S. "Using Particles to Sample and Control Implicit Surfaces", *Computer Graphics*, 1994, pp.269-277.

[8] Bloomenthal, J.,"Polygonization of implicit surfaces", *CAGD*,5, 1988, pp.314-355,

[9] Shen J., Thalmann, D. " Interactive Shape Design Using Metaballs and Splines ", *Proc. Eurographics Workshop on Implicit Surfaces '*95 , Grenoble, pp.187-196

[10] Magnenat Thalmann, N., Thalmann, D. "Digital Actors for Interactive Television", *Proc. IEEE*, Special Issue on Digital Television, Part 2, July 1995, pp.1022-1031.

[11] Noser, H., Çapin, T., Pandzic, I., Magnenat Thalmann, N., Thalmann, D. "Playing Games using the Virtual Life Network", *Proc. Alife '96*, Nara, Japan, 1996.

[12] Pandzic, I., Çapin, T. Magnenat Thalmann, N., Thalmann, D. "VLNET: A Networked Multimedia 3D Environment with Virtual Humans", *Proc. Multi-Media Modeling MMM '95*, Singapore, 1995, pp.21-32.