Animating Complex Hairstyles in Real-Time

Pascal Volino
MIRALab, Univ. of Geneva
CH-1211, Geneva, Switzerland
+41 22 379 10 76
pascal@miralab.unige.ch

Nadia Magnenat-Thalmann
MIRALab, Univ. of Geneva
CH-1211, Geneva, Switzerland
+41 22 379 77 69
thalmann@miralab.unige.ch

ABSTRACT
True real-time animation of complex hairstyles on animated characters is the goal of this work, and the challenge is to build a mechanical model of the hairstyle which is sufficiently fast for real-time performance while preserving the particular behavior of the hair medium and maintaining sufficient versatility for simulating any kind of complex hairstyles.

Rather than building a complex mechanical model directly related to the structure of the hair strands, we take advantage of a volume free-form deformation scheme. We detail the construction of an efficient lattice mechanical deformation model which represents the volume behavior of the hair strands. The lattice is deformed as a particle system using state-of-the-art numerical methods, and animates the hairs using quadratic B-Spline interpolation. The hairstyle reacts to the body skin through collisions with a metaball-based approximation. The model is highly scalable and allows hairstyles of any complexity to be simulated in any rendering context with the appropriate tradeoff between accuracy and computation speed, fitting the need of Level-of-Detail optimization schemes.

Categories and Subject Descriptors: I.3.7 [Three-Dimensional Graphics and Realism]: Animation. Virtual Reality; I.6.3 [Simulation and Modeling]: Applications. General Terms: Algorithms. Keywords: Real-time animation, mechanical simulation, hair modeling, virtual characters.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VRST '04, November 10-12, 2004, Hong Kong.
Copyright 2004 ACM 1-58113-907-1/04/0011...$5.00.

1. INTRODUCTION
Modeling and rendering hair on virtual characters always remains a challenge. Its nature of intricate arrangement of a huge number of strands (often more than 100 000) provides to the medium a complex behavior that depends not only on the elastic properties of the strands, but also on the way they interact through friction. This behavior is highly dependent on the type of hairstyle (straight or curly, long or short) and condition (stiff or soft, dense or sparse, dry or wet...). Techniques may consider any compromises between simulating individually each strand and simulating a volume medium representing the complete hairstyle.

1.1. Overview of Hair Simulation Techniques
Hair modeling has started along the emergence of computer-generated virtual characters. The first models were explicit, and considered the modeling and the rendering of each individual strand. These are illustrated by the contributions of Daldegan et al [9] for modeling and rendering, Rosenblum et al [31], Kurihara and Anjyo et al [1] for strand-based animation.

However, the explicit model had serious limitations resulting from the large amount of hair strands to be considered, resulting in large computation times unsuitable for real-time applications. Reducing this complexity and the number of degrees of freedom of the model is essential for fast computation. The first idea is to consider that hairs of neighboring locations share similar shapes and mechanical behaviors. In many models, such hairs are grouped into wisps, sometimes also called clusters. This eases the work of the hair designer as well as the computation required for hair animation. This technique was defined by Watanabe et al [34], and has been frequently used since with many variations. Examples are the guide hairs with collisions and interpolation of Chang et al [6], the multi-layer approach with cluster collisions of Plante et al [29], and some variations from Chen et al [7] and Yang et al [35]. Another evolution is to replace the hairs by approximate surfaces (strips or patches), as done by Koh et al [23] [24], or even volumes that can easily be modeled as polygonal meshes, such as the thin shell approach of Kim et al [21]. Animation is done by specific mechanical models associated to these representations, which also include specific methods for handling collisions efficiently as described by Lee et al [27].

Combining various approaches can lead to Level-of-Detail methods where the representation is chosen depending on the current requirement of accuracy and available computational resources. A good example developed by Bertails et al [4] is based on wisps tree, and a similar approach from Kim et al [22] is based on multiresolution clusters. Advanced Level-of-Detail methods also include combination of strands, clusters and strips modeled with subdivision schemes and animated using advanced collision techniques, as developed by Ward et al [32] [33].

In another kind of approach, volume hair models construct a representation of the hair based on a vector fields representing the orientation of the hair strands. These approaches have been exploited for hair design by Hadap et al [16], like a similar model from Yu [36]. A major difficulty of volume hair models is the difficulty to connect them to the actual mechanics of hair strands. An attempt based on explicit strand mechanics has been carried by Hadap et al [18], but the resulting computation times were incompatible with interactive applications.

In the specific context of real-time applications, cluster and strip models are good candidates. Fast strip or guide hair animation can be obtained through simplified articulated body mechanics, such
Our idea is to use Free-Form Deformations (FFD) to animate the hair in real-time. Free-Form deformations are widely used for shape design and deformation [13] and recent developments also include mechanical deformations [10].

Principles of the method

Our method works as follows (Figure 2): During preprocessing, a lattice is defined around the head and the initial hairstyle. A mechanical model is then constructed to provide the lattice with a mechanical behavior related to the hair contained in it. This mechanical model is based on a particle system constructed on the lattice nodes. A particular kind of particle interaction has been developed to represent the mechanical behavior of a hair strand on the lattice. This model can be decimated at will to obtain the best compromise between accuracy and computation speed. Collisions act directly on the lattice nodes through repulsion forces from a simple metaball representation of the head and shoulders. During computation, the lattice is animated using state-of-the-art simulation methods for integrating the mechanical behavior of particle systems. The current positions of the hair strands or any other rendered features are finally computed at rendering time using linear or quadratic B-Spline interpolation from the current position of the lattice nodes.

Benefits

At the first glance, it seems pretty unnatural to map hairstyle deformation on a simple cubic lattice rather than using an interpolation scheme more fitted to the actual topology of the hairstyle (for instance, interpolating between a reduced set of animated guide hairs). However, this method allows to take advantage of several benefits:

- The cubic lattice defines a true volume around the skull, and any location inside this volume can be computed using very simple interpolation formula (fast interpolation of any feature during animation).
- Computing lattice coordinates of any location is also fast and straightforward (for use in interactive hairstyle editing).

Decoupling the topology of the animated interpolation structure from the actual hair topology also have good benefits:

- Limited reliance on the strand nature of the hair, which is only used during preprocessing for constructing the mechanical model representing hair mechanics. The resulting Free-Form interpolation can be directly applied on strands as well as on clusters, strips and even volume representations of the hair, giving a highly versatile model suitable for all rendering contexts.
- Total freedom for hairstyle design, as hairs can be long or short, curled or straight, and even contain specific features such as ponytails without any particular adaptation of the model. This is quite impossible with current wisp-based animation systems that require wisp shapes to be in a mechanically consistent state (usually straight) only influenced by collisions.

- Good scalability, by allowing complete control of the complexity of the mechanical model (lattice size and number of interactions) independently from the complexity of the initial hairstyle. Realism of the hair animation is maintained to an according extent by the mechanical interactions in the lattice that are constructed to model the mechanical behavior of hair strands.

Using our system, the hairstyle can be designed in its natural equilibrium state on the head using any hairstyle creation software, which is much easier to manage for the designer. Whatever the input, the lattice and its mechanical model are automatically constructed at any resolution suitable for the performance requirements.

Figure 2: Workflow of the hair animation process: The hairstyle is designed (B) from a model (A). A lattice is then build from the hair strands with a simplified mechanical model (C). During animation, the lattice is mechanically deformed (D) and hair features are interpolated from the lattice (E) for real-time animation and rendering.

Section 2 describes the principles of our simulation scheme, whereas Section 3 details the mechanical model and Section 4 the geometric Free-Form interpolation method. Results and further considerations are detailed in Section 5.

2. LATTICE-BASED FFD ANIMATION

In a typical real-time animation system, the body animation is usually carried out by animating a skeleton through various high-level methods, such as prerecorded animation, real-time tracking or smart autonomy algorithms. A transformation matrix is associated to each skeleton component, and expresses its current position in world coordinates. The body skin is deformed using these matrices. Interpolation methods may eventually deform the skin around the joints. We focus our attention on the motion of the head, described at any time by a transformation matrix \( R \) and speed matrix \( R' \) (element-wise derivative of \( R \) against time).
2.1. The Lattice Model

We construct a 3D lattice around the head that includes all the hair. As shown in Figure 4, this lattice is defined by the initial position vector \( \mathbf{P}^0 \) containing the positions \( \mathbf{p}_i^0 \) of all its nodes. During animation, the lattice is moved according to the head motion. The current rigid-motion positions \( \mathbf{R} \mathbf{p}_i \) and velocities \( \mathbf{R'} \mathbf{p}_i' \) of the lattice nodes define the rigid motion pattern of the hair, in which the hair follows the head motion without any deformation (Figure 3).

In our approach, the lattice is however deformed by mechanical computation. For this, we define the current deformed position vector \( \mathbf{P}' \) and velocity vector \( \mathbf{P}'' \) of the lattice containing the current positions \( \mathbf{p}_i' \) and speeds \( \mathbf{p}_i'' \) of the lattice nodes (Figure 4). Their evolution is ruled by mechanical computation iterations that use the current values of \( \mathbf{R} \mathbf{p}_i^0 \) and \( \mathbf{R'} \mathbf{p}_i^0 \) as equilibrium states.

This mechanical model, aimed at providing the lattice with the volumic behavior of the hair contained in it, is constructed during preprocessing.

\[
\begin{align*}
\frac{\partial \mathbf{f}_i}{\partial \mathbf{p}_j} &= w_i \frac{\partial \mathbf{p}_j}{\partial \mathbf{e}} \quad \text{and} \quad \frac{\partial \mathbf{f}_i}{\partial \mathbf{p}_j'} &= w_i \frac{\partial \mathbf{p}_j'}{\partial \mathbf{e}} \quad (2)
\end{align*}
\]

Weight vectors are typically constructed using the linear interpolation coefficients for defining the virtual locations on which the forces are exerted in the lattice. Among them, we use linear lattice springs of which can be used to create a viscoelastic spring force (elasticity \( k \) and viscosity \( q \)) between two points anywhere in the lattice volume, using a behavior law defined as follows:

\[
\begin{align*}
\mathbf{\sigma} &= (-k|\mathbf{e}|-\mathbf{q}(\mathbf{e}^T \mathbf{e}')) \mathbf{e} \quad \text{with} \quad \mathbf{e} = |\mathbf{e}| \mathbf{e} \\
\frac{\partial \mathbf{\sigma}}{\partial \mathbf{e}} &= -k \mathbf{e}^T \quad \text{and} \quad \frac{\partial \mathbf{\sigma}}{\partial \mathbf{e}} = -\mathbf{q} \mathbf{e}^T \quad (3)
\end{align*}
\]

We also use linear lattice attachments that attach any point of the lattice volume to a particular position with a viscoelastic force, defined as follows:

\[
\begin{align*}
\mathbf{\sigma} &= -k(\mathbf{e} - \mathbf{e}_a) - \mathbf{q}(\mathbf{e}') \\
\frac{\partial \mathbf{\sigma}}{\partial \mathbf{e}} &= -k \mathbf{I} \quad \text{and} \quad \frac{\partial \mathbf{\sigma}}{\partial \mathbf{e}} = -\mathbf{q} \mathbf{I} \quad (4)
\end{align*}
\]

The rest position \( \mathbf{e}_a \) of any stiffener is precomputed from the initial positions of the lattice nodes \( \mathbf{p}_i^0 \) and the current skeleton transformation matrix \( \mathbf{R} \) (which has constant scale) as follows:

\[
\mathbf{e}_a = \mathbf{R} \sum_j w_j \mathbf{p}_j^0 \quad (5)
\]

2.2. An Efficient Model using Lattice Stiffeners

Different approaches are possible for designing this mechanical lattice deformation model. However, spring-mass approaches still seem to be the best candidate for designing really fast mechanical models.

Restricting the model to springs directly linking the lattice nodes imposes force directions ruled by the lattice orientation. This is too inaccurate to precisely model the real effect of elastic hair segments that follow very precise arbitrary directions. The only way to allow a spring to represent accurately a hair segment is to restrict the model to springs directly linking the lattice nodes.

A possible solution is to create additional intermediate nodes inside the lattice elements. Such approaches are usually combined to adaptive subdivision schemes. This would however complicate the FFD interpolation procedures, and the benefit of a reduced number of degrees of freedom would be lost as well.

Lattice stiffeners

We have solved this problem by creating lattice stiffeners, a kind of mechanical interaction model that acts on a weighted sum of lattice nodes rather than on individual nodes only (Figure 4). They are defined by a behavior law \( \mathbf{\sigma}(\mathbf{e}, \mathbf{e}') \) that relates particle forces \( \mathbf{f}_i \) to particle positions \( \mathbf{p}_i \) and velocities \( \mathbf{p}_i' \), as well as a particle weights \( w_i \) that relates the influence of the law on each lattice node, as follows:

\[
\mathbf{f}_i = w_i \mathbf{\sigma} \quad \text{with} \quad \mathbf{e} = \sum_j w_j \mathbf{p}_j \quad \text{and} \quad \mathbf{e}' = \sum_j w_j \mathbf{p}_j' \quad (1)
\]

Lattice stiffeners are quite suited for being integrated with implicit methods, as their force derivative contribution to the Jacobian matrix of the system are simply computed as follows:

\[
\frac{\partial \mathbf{f}_i}{\partial \mathbf{p}_j} = w_i \frac{\partial \mathbf{p}_j}{\partial \mathbf{e}} \quad \text{and} \quad \frac{\partial \mathbf{f}_i}{\partial \mathbf{p}_j'} = w_i \frac{\partial \mathbf{p}_j'}{\partial \mathbf{e}} \quad (2)
\]
model additional external forces such as gravity and aerodynamic effects. This is detailed in Section 3.2.

- The Collision effects against the body, which prevents hair penetrating through the body skin. Their representation is detailed in Section 3.3.

3.1. The Mechanical Model of the Hair

The mechanical model of the hair is defined as a sum of linear viscoelastic lattice springs relating the elasticity of each individual hair strand segment in the lattice model. Their viscoelastic parameters correspond to those of the modeled hair. The attachments of the hair extremities to the skull are modeled by stiff viscoelastic lattice attachments, which are positioned exactly at the end of each hair. The resulting mechanical model is shown in Figure 6.

In order to reduce the complexity of the model, we resample each hair during the model construction as segments defined by the intersection points of the hair line on the lattice boundaries for which the crossing angle is above 45°, as shown in Figure 5. This limits the number of created lattice springs whatever the discretization of the initial hair curve, as well as the number of particles involved in each lattice springs.

![Figure 5: Construction of the lattice springs (green lines) of a hair strand (yellow line) with the proposed rediscretization (blue dots) on the lattice. The lattice nodes along the line (gray dots) are affected.](image)

Each hair also contributes to the mass of the lattice nodes, as shown in Figure 6. The mass of a hair segment is shared on the corresponding lattice nodes according to the linear interpolation coefficients of its extremities. Nodes not involved in any hair segment are trimmed away from the simulation.

![Figure 6: From a hairstyle (left), the corresponding mechanical model is built using lattice stiffeners (center), and the mass of the hair strands is distributed on the lattice nodes (right) (darker is heavier).](image)

While this model makes a good approximation of the mechanical behavior of the hair on the lattice model, its computation time is still much too high because of the high number of hairs. We therefore carry out a simplification of the model by decimating the redundant lattice stiffeners (hair springs as well as skull attachments), as shown in Figure 7. This simplification is carried out by evaluating, for all couples of stiffeners sharing common vertices, the "mechanical error" resulting from merging one of the stiffeners to the other. Error and merging computations are based on dot product operations on the weight vectors, which evaluate the "synergy" between stiffeners (their "parallelism" in the weight space). The merging operations that minimize the error are carried out until the expected number of stiffeners remain. We avoid quadratic decimation search times by constructing the adequate temporary data structures.

![Figure 7: The decimation process: From the strands of the hairstyle (up-left) is constructed the initial model (up-right) containing roughly 14200 springs and 3600 attachments. The model can then be decimated at will, for example 800 springs and 200 attachments (down-left), or 200 springs and 50 attachments (down-right).](image)

3.2. The Ether Model

While this properly gives an adequate behavior of the lattice nodes inside the hairs, nodes surrounded by lattices empty of any hair are quite underconstrained and may exhibit erratic behavior. To limit this, we handle the whole lattice as an "ether medium" that has its own stable behavior. Besides adding to all lattice nodes an ether mass, we add viscoelastic ether forces relating each node to its rest position defined by rigid head motion. They are modeled as one-particle linear viscoelastic lattice attachments. They have a weak constant elasticity parameter which pull back each node to its equilibrium position, as well as a constant viscosity parameter for limiting oscillatory behavior.

Besides limiting erratic behavior of the lattice nodes, adjusting parameters of ether forces are a good way to control the global stiffness of the hair design (for example simulating designs with hair gel), as well as roughly simulating aerodynamic damping effects. Ether forces are also the key factors for ensuring robustness of a hair model which should return close to its initial posture after whatever mechanical or not-so-mechanical body motion.

The ether also supports additional external forces, such as aerodynamic effects. Air drag and wind is simulated by our system by assigning to each lattice node a viscous reactivity parameter representing the drag created by the hair surrounding this node. These factors are actually tensors which are precomputed by distributing the anisotropic drag contributions of each hair segment to the neighboring lattice nodes (similarly to how the hair mass is distributed). Using a global viscosity parameter $q$ (air drag per unit hair length), the contribution tensor $Q$ of a hair segment $\mathbf{t}$, which only creates drag perpendicularly to the segment direction, is computed as follows:
\[ Q = a \mathbf{I} (1 - \mathbf{e} \mathbf{e}^T) \quad \text{with} \quad \mathbf{e} = \mathbf{e}^T \mathbf{e} \] (6)

This aerodynamic model can successfully model the drag created by airflow with very minor impact on the computation time. For high wind situations, some additional wind speed perturbations are welcome for simulating turbulence (Figure 8).

3.3. Handling Collisions
Collision detection is usually one of the most tedious computational tasks, particularly for time-critical applications such as real-time simulations. In the case of hair animation, this complexity results from the huge hair geometry as well as the complex shape of the body surface.

Collisions between individual hair strands are not really an issue when using volume deformations, as volume deformation continuity of FFD preserves the local relative positions between the deformed hair strands. The only issues may concern self-intersection of the FFD lattice in case of huge deformations, which in practice does not occur so as to produce disturbing animation effects.

The real issue is actually the detection and response of collisions between the hair and the body surface, which should prevent hair from entering the skin surface of the body.

Traditional collision approaches would consider computing the intersection between the hair geometry and the polygonal mesh that describes the body surface, possibly using optimizations based on bounding volumes, subdivision and incremental evaluation, which in any case remains a tedious task. We overcome this difficulty by replacing the body surface description by an approximate model which defines a repulsion force exerted on the lattice nodes from an analytically-defined volume energy potential roughly describing the body volume. Unlike surface-based approaches, such a scheme ensures robust collision handling by pushing out even deeply penetrating points unambiguously.

We have chosen 6th-order polynomial metaballs to perform this modeling. Metaballs are widely-used primitives for implicit surface modeling [5]. They have a well-defined finite radius of influence and are good additive primitives for modeling full volumes with adequate continuity properties, while evaluation of their repulsion forces (gradient of their energy potential) can be computed efficiently. 6th-order also offer null derivative (Jacobian) on the metaball limit, offering suitable continuity properties for use with implicit integration methods.

Yet, for real-time applications, we cannot afford to compute the influence of more than a few metaballs for each lattice node. However, an accurate model of the head and shoulders of a body may require several tens of metaballs depending on the required accuracy. Rather than working with a huge number of metaballs, we have chosen to customize the parameters (radius and base potential) of a reduced number of metaballs for each lattice node. Hence, a given node "sees" its own custom modeling of the skin with a suitable accuracy relatively to the relative position of the node along the body skin (Figure 9).

The customization scheme works as follows:
- If a lattice node is initially positioned outside the mesh, the metaball radius is chosen so as to prevent it from coming nearer to the mesh than a given "hair thickness" distance.
- If a lattice node is inside the mesh, the metaball radius is chosen so as to prevent it from going further inside the mesh. We prefer this option rather than completely blocking the node, in order to allow better tangential sliding of the hair on the skin surface.

Experimentally, this approach has shown to prevent quite successfully collisions from occurring against the body skin using a very small number of metaballs. For instance, a typical skull can be represented using one or two metaballs, whereas six to ten metaballs would be required when the hair may touch the shoulders and chest as well (Figure 10), depending on the wanted collision accuracy. Its robustness also allows this method to cope with any physical or not-so-physical head motions.

Metaball models do also nicely deform with simple displacement of their centers. Metaballs representing the shoulders and upper trunk are attached directly to the skeleton of these body parts rather than to the head, and our method is therefore applicable for deformable characters (Figure 11).
4. LATTICE FFD FOR HAIR ANIMATION
As the lattice is deformed during animation, another issue is to
recompute the current position of each hair features for each
frame. Depending on the selected rendering techniques and
optimizations, these features may either be individual hair
segments, or larger primitives such as hair strips or groups.
Different kinds of interpolations allow this to be performed with
various degrees of continuity. They define how the weight
coefficients should be computed from the initial position of the
feature relatively to the undeformed lattice. A good review of
interpolation curves and surfaces is described in [11].

The most continuous interpolations are based on Bezier curves.
However, these are impractical as the weight vector is not sparse
(the interpolated point position depends on the positions of all
the nodes of the lattice), and therefore very inefficient to compute.
For the best compromise between continuity and computation
speed, we have selected quadratic B-Spline curves, which offer
second-order interpolation continuity. For 3D interpolation, each
interpolated point is a linear combination of the 27 nearest nodes
of the lattice. In our adaptation, we use linear extrapolation to
handle border nodes so as to decrease deformations for points
located outside the lattice (Figure 12).

In applications where interpolation is really time-critical, linear
interpolation still remains a good candidate. The resulting first-
order continuity still looks acceptable if the lattice deformations
are not too large, and the computation is roughly three times faster
as each interpolated point is only a linear combination of the 8
nearest lattice nodes.

In our hair interpolation scheme, we precompute and store for
each hair feature that needs to be rendered the sparse vector
containing the weights corresponding to all lattice nodes. Then,
during rendering, the current interpolation is simply computed by
a weighted sum of the current lattice node positions.

We can take advantage of the interpolation to enhance the
attachment of the hair on the skull through rigid motion. For each
interpolated feature, a deformation coefficient $\delta$ is defined which
is a blending coefficient between the motion defined by the rigid
head motion ($\delta = 0$) and the motion defined by the interpolated
lattice position ($\delta = 1$). In our implementation, we vary
progressively the deformation coefficient from 0 to 1 from the root
to the extremity of each hair according to the following expression:

$$\delta = 1 - \exp\left(\frac{\lambda^2}{\lambda_0^2}\right)$$  (7)

Where $\lambda$ is the distance of a hair point from the root along the hair
curve, $\lambda_0$, which is the "typical" hair length from the root that does
not deform significantly, acts as a parameter for defining the
bending stiffness of hair near its root (Figure 13). This also
removes the artifacts resulting from the imperfections of the
mechanical attachment of the hair roots to the moving skull.

There are many other ways to alter the deformation coefficient on
different regions of the hair, for example for differentiating stiff
hairs from soft ones, or introducing some randomness in the
motion of various hair fibers.

5. RESULTS
We have implemented the computational algorithms of our system
in standard C++ with standard floating-point math. The system
was tested on a 3GHz Pentium4 PC with a nVIDIA Quadro 980
XGL graphics card and 512MB of memory running Microsoft

Hair rendering is also critical issue for achieving realistic display
in real-time. The beauty of the hair mainly lies in the way it
interacts with the light. A good example of lightning model is
studied by Marschner et al [28]. Our implementation is mainly
based on real-time rendering of strands and textured strips [33]
using the anisotropic lightning model described by Kajiva et al
[20]. This model was implemented by bypassing the standard
OpenGL Texture & Lightning pipeline with our own illumination
model implemented using nVIDIA’s Vertex Shaders. We
integrate this rendering in a Level-of-Detail scheme by adapting
dynamically the number of rendered strands and strips according
to the required visual accuracy.

5.1. Performances
We have tested the hair model shown in Figure 8 with various
complexities of the mechanical representation. The high-end
mechanical model, which represents a very good simulation
accuracy, contains 200 attachments and 800 springs built on a grid
containing 1000 lattice nodes. The low-end model, still fairly
accurate, only contains 50 attachments and 200 springs built on
125 lattice nodes. Both models react to collisions with the head
and the shoulder collisions using 7 metaballs. Models with no
attachments and springs and no metaballs and relying only on
ether stiffness, which may still be used for certain hair designs
(short or stiff hair) or low accuracy requirements, are also tested
for comparison.

The following table gives the time necessary for the mechanical
computation of one frame (1/25 s) of the animation. The
resolution of the implicit mechanical model uses 16 iterations of
the Conjugate Gradient method. More iterations may be necessary
for longer frame times or particular stiff hair models.

<table>
<thead>
<tr>
<th>Lat.Size</th>
<th>Mech.Mod</th>
<th>0 Att.</th>
<th>0 Spr.</th>
<th>50 Att.</th>
<th>200 Spr.</th>
<th>100 Att.</th>
<th>200 Spr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10x10x10</td>
<td>0 Meta.</td>
<td>1.1 ms</td>
<td>5.2 ms</td>
<td>10.1 ms</td>
<td>13.8 ms</td>
<td>21.4 ms</td>
<td></td>
</tr>
<tr>
<td>5x5x5</td>
<td>0 Meta.</td>
<td>0.1 ms</td>
<td>0.2 ms</td>
<td>2.1 ms</td>
<td>3.8 ms</td>
<td>7.6 ms</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Computation time (milliseconds) of one animation
frame using different mechanical model complexities and
lattice sizes.
Our implementation also needs approximately 2.5 ms for interpolating 10,000 features using quadratic B-Spline FFD, and only 1.0 ms using linear FFD. Adequate parametrization allows most hairstyles to be mechanically simulated within 10 milliseconds for one frame, which is 1/4 of the 40 milliseconds available (25 frames per second). This leaves 30 milliseconds for carrying out the animation of the face, body and other features of the scene, as well as rendering.

Figure 14: Various hairstyles animated by our real-time simulation system.

For most models, rendering is now the most time-consuming task of the whole real-time process. While the best rendering qualities are obtained with our system through explicit rendering of the hair strands with lighting and texture, this consumes between 50 and 100 milliseconds for about 10,000 strands. This is roughly the performance obtained for the rendered hairstyles shown in Figure 14. Better performance is obtained by reducing the number of rendered features and using textured strips instead of hairs. Rendering 500 textured strips roughly consumes 20 milliseconds in our implementation.

Figure 15: Level-of-Detail: 200 ms for animating and rendering a frame with high-accuracy (left), and 40 ms with low-accuracy (right).

We have taken advantage of the huge performance difference between models of various accuracies and rendering methods for implementing a Level-of-Detail animates scheme that simulates a given hairstyle by switching between several mechanical models depending on the required accuracy (related to on-screen head size and character motion speed). This allows total frame computation times between roughly 200 milliseconds (10 x 10 x 10 lattice, model containing 1000 springs and 250 attachments, 10,000 rendered textured hair strands with quadratic interpolation, textured body animation) and 40 milliseconds (5 x 5 x 5 lattice, 200 springs and 50 attachments, rendering 200 textured hair strips with linear interpolation, low-quality body animation) (Figure 15). Faster times may still be obtained is the hair is approximated as rigid (lowest detail level).

5.2. Perspectives

The major benefit of this approach is its scalability, and its versatility. First, the approach allows a clear distinction between the mechanical model that animates the lattice and the actual objects that are deformed by the lattice during the rendering. This greatly eases the task of combining efficient simulation with complex hair representations, as well as designing level-of-detail schemes that can act independently on the mechanical simulation aspect and on the rendering aspect. This high versatility also allows the simulation of any hairstyle directly from the output of hairstyle design system without any specific handling. Hence, long flowing hair can be animated along with short and stiff hair using the same models. It is quite easy to extend the model to support extra original features (weighted hair knots, threads, and even cloth) through the simple addition of the corresponding mechanical behaviors in the particle system constituted by the lattice nodes.

This method still has plenty of room for evolution. The algorithmic simplicity of the model also turns it into a good candidate for hardware implementations. We could also benefit from the low number of mechanical degrees of freedom to replacing mechanical simulation by fast animate-by-example methods such as the one described by Grzeszczuk et al [14]. We also plan to take advantage of the versatility to create real-time hairstyling systems based on mechanical simulation. We are still looking forward for many new developments to the exciting field of animated virtual characters.

Figure 16: Animating a large diversity of hairstyles.

Acknowledgements

We are grateful to all participants to this research for their contributions and advices, as well as to Nedjma Cadi and Cristiane Luible for their high-quality creative work. This project is funded by the Swiss Fonds National pour la Recherche Scientifique.

Figure 17: Easy simulation of real-world hairstyles.

REFERENCES

