Modeling Hair Using Level-of-Detail Representations

Kelly Ward Ming C. Lin Joohi Lee Susan Fisher Dean Macri
University of North Carolina at Chapel Hill Alias|Wavefront Pixar Studio Intel Corporation
http://gamma.cs.unc.edu/HSLOD/

Abstract: *We present a novel approach for modeling hair using level-of-detail representations. The set of representations include individual strands, hair clusters, and hair strips. They are represented using subdivision curves or surfaces, and have the same underlying base skeleton to maintain consistent high-level physical behavior when a transition between different levels-of-detail occurs. This framework supports automatic simplification of dynamic simulation, collision detection, and graphical rendering of animated hair. It also offers flexibility to balance between the overall performance and visual quality, and can be used to model and render different hairstyles. We have used these level-of-detail representations to animate various hairstyles and obtained noticeable performance improvement, with little loss in visual quality.*

Keywords: Geometric Modeling, Hair Modeling, Level-of-Detail Algorithms.

1 Introduction

The ability to model human features has become an essential aspect of 3D graphics for modeling avatars in virtual environments, virtual humans in computer games, and human characters in animated films. A human head can have over $100,000$ individual strands of hair. Animating hair in real-time is a challenging problem due to the high number of primitives required to model hair accurately and realistically. It is also difficult to achieve stable and robust simulation of hair dynamics as well as interactions among the hair and between the hair and the body for different types of hairstyles.

The current commercial renderers, such as Pixar's RenderMan and other proprietary software used in movies like *Monsters, Inc.* and *Final Fantasy*, can generate beautiful and realistic appearances for hair and fur. However, to the best of our knowledge, these systems do not offer interactive performance for either animation or rendering of hair. Modeling, styling, simulating and animating hair remains a slow, tedious and often painful process for animators. A few existing real-time algorithms [Koh and Huang 2001] or hardware accelerated rendering and shading [NVIDIA 2001], on the other hand, often do not yield realistic hair renderings or are limited to certain hairstyles (e.g. short, straight hair). One of the major bottlenecks in achieving real-time simulation of moving hair is collision detection among hairs and between the hair and the body. This is often the dominating factor in terms of the overall computational cost [Plante et al. 2001].

Main Contribution: In this paper, we present a novel, unified framework for modeling hair based on level-of-detail (LOD) representations. We use a series of subdivision surface patches [Schröder and Zorin 1998] for modeling the least significant layers of hair, hair clusters represented as subdivision surfaces of variable thickness for modeling the intermediate layers, and subdivision curves for simulating the most visible and highest-resolution individual hair strands. Two hairstyles modeled and simulated using a combination of these representations are shown in Fig. 1 and Fig. 2. Our algorithm combines this set of LOD representations to simulate moving hair, perform collision detection, and accelerate graphical rendering. It automatically switches between different approximations of varying fidelity, depending on the user specified screen-space error tolerance, viewing distance, visibility, hair motion and other application dependent factors. Overall, our framework offers several advantages:

- **Unified representations** based on the subdivision framework and the "base skeleton" representation;

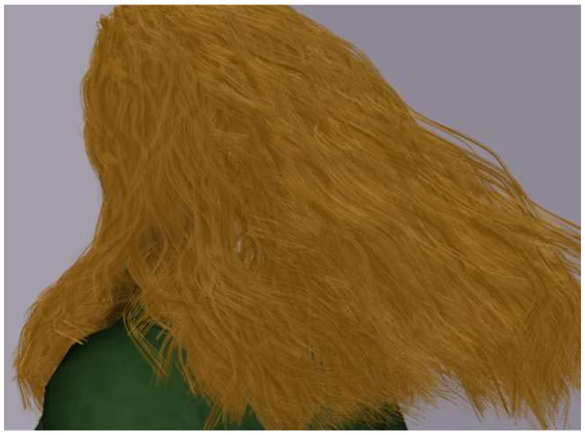
Figure 1: *Long, curly red hair blowing in the wind.*

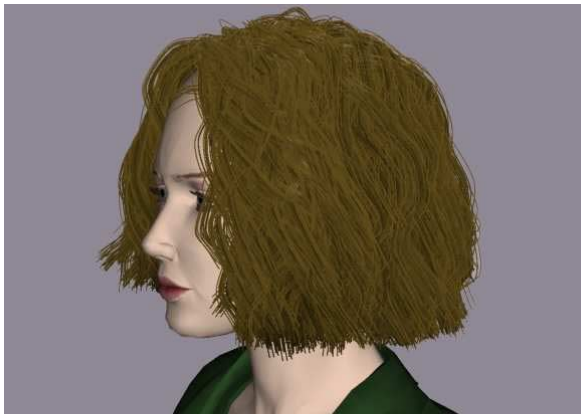
Figure 2: *Short, wavy brown hair.*

- **Automatic simplification** of both geometry and physics for hair animation and rendering;
- **Computational efficiency** in the overall dynamic simulation, collision detection, and graphical rendering;
- **Flexibility** in achieving the desired balance between simulation speed and visual fidelity;
- **Generality** in terms of modeling many different types of hair: short vs. long, straight vs. wavy, thin vs. thick, fine vs. coarse.

Using these level-of-detail representations for modeling hair, we observed noticeable overall performance improvement with little degradation in visual appearance of the simulation.

Organization: The rest of the paper is organized as follows. Section 2 gives a brief survey of related work. Section 3 describes the three basic model representations of hair based on the subdivision framework and the base skeleton. The dynamics model and our collision detection algorithm using the LOD representations are described in Section 4 and Section 5 respectively. In Section 6, we describe techniques to render hair using the proposed level-of-detail representations. The criteria for automatically switching and selecting LOD representations are outlined in Section 7. Section 8 highlights the results of our implementation, and analyzes its performance. Finally, we suggest several areas for future work.

2 Related Work

Our work is built upon a large body of knowledge and concepts from several different areas, including hair modeling, rendering and animation, multiresolution representations, and simulation acceleration techniques. We synthesize together many key ideas from different areas, improve upon several known algorithms for simulating and rendering hair, and propose a new approach for hair modeling based on the subdivision framework and the base skeleton.

2.1 Hair Modeling

Modeling hair has been an active area of research in computer graphics and numerous approaches have been proposed to address this problem [Magnenat-Thalmann et al. 2000; Hadap and Magnenat-Thalmann 2001; Yu 2001; Kim and Neumann 2002]. Some fundamental techniques were presented to model the motion of individual hair strands in [Anjyo et al. 1992; Kurihara et al. 1993; Daldegan et al. 1993], with each strand of hair represented as a series of connected line segments and the shape of the hair determined by specifying the desired angles between segments. Forces are applied to the control points of the line segments to simulate the hair motion. To reduce the overall computation time, strands of hair that are near each other or move in a similar fashion, are bundled together as a group or as a *wisp* [Kurihara et al. 1993]. Using a similar philosophy, individual strands of hair are grouped together as "wisps" for animating long hair, each modeled using a spring-mass skeleton and a deformable envelope [Plante et al. 2001]. A similar approach is used for interactive hairstyling [Chen et al. 1999; Xu and Yang 2001]. Adaptive guide hairs were used in [Chang et al. 2002] to add more detail to overly interpolated regions. Using guide strands involves animating a few strands and the dynamics of the remaining strands are interpolated from these guides.

None of these techniques, though, can perform hair animation or rendering in real-time. Recently, a thin shell volume [Kim and Neumann 2000] and 2D strips [Koh and Huang 2000; Koh and Huang 2001] have been used to approximate groups of hair. Such techniques enable real-time hair simulation. However, the resulting simulation lacks a realistic, voluminous appearance of the hair. Techniques for real-time rendering of fur and hair that exploit graphics hardware were presented in [Lengyel 2000; Lengyel et al. 2001; NVIDIA 2001]. However, these techniques do not work well or are not applicable for rendering long, wavy or curly hair.

2.2 Model Simplification

Model simplification algorithms, such as automatic generation of geometric level-of-detail (LOD) representations and multiresolution modeling [Schröder and Zorin 1998] techniques, have been proposed to accelerate the rendering of complex geometric models. A recent survey on polygonal model simplification is presented in [Luebke 2001]. A generic framework for selecting and switching between different geometric levels-of-detail (LODs) to attain a nearly constant frame rate for interactive architectural walkthroughs was introduced in [Funkhouser and Séquin 1993].

2.3 Simulation Level-of-Detail

The use of levels-of-detail has been extended to motion modeling and dynamic simulation as well. Simulation levels-of-detail (SLOD) are used to simplify or approximate the dynamics in a scene, similar to the way that geometric LODs are used to simplify a complex model.

Carlson and Hodgins explored techniques for reducing the computational cost of simulating groups of legged creatures when they are less important to the viewer or to the action in the virtual world [Carlson and Hodgins 1997]. In [Perbet and Cani 2001], levels-of-detail, including 3D geometry, volumetric textures and 2D textures, are used to animate and render prairies in real-time. SLODs have also been proposed for the automatic dynamics simplification of particle systems [O'Brien et al. 2001].

Other types of simulation acceleration techniques, such as view-dependent dynamics culling [Chenney and Forsyth 1997] and Neuro-Animator [Grzeszczuk et al. 1998], have also been investigated to reduce the total computational costs for simulating a large, complex dynamical system.

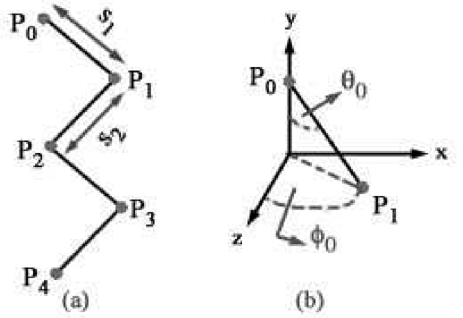

Figure 3: Basic Physics Models. *(a) The base skeleton model; (b) The parameters that define the style of hair.*

3 Model Representations

Our approach uses three novel representations based on the subdivision framework and a base skeleton to create both discrete and continuous levels-of-detail for hair. They are *strips*, *clusters*, and *individual strands*.

3.1 Design Consideration

Although the set of proposed LOD representations may appear to be intuitive and simple, it is carefully designed and chosen. We have adapted the use of the subdivision framework. The subdivision framework can model different hairstyles as effectively as NURBS, quickly perform adaptive dynamic tessellation, and most of all can potentially take advantages of new graphics hardware for interactive rendering of curve primitives. (More detail will be given in section 6.)

The use of the base skeleton is intentionally selected to maintain a global, consistent, macroscopic physical behavior, as LOD switches take place. This choice helps to drastically simplify many transition difficulties typically present during LOD switching. It automatically reduces a fairly high degree-of-freedom dynamical system down to a lower degree-of-freedom dynamical system, without any extra expensive computations other than performing the LOD switching tests (to be described in section 7).

3.2 Subdivision Representations

Subdivision curves and surfaces have been chosen as the underlying geometric representation for all LODs in our hair modeling framework because of their scalability and uniformity of representation [Schröder and Zorin 1998]. The subdivision process creates smooth curves and surfaces through successively refining a curve or mesh of control points. Defining the levels of successive refinement can control the smoothness of the resulting surface or curve. This is used to generate adaptive, continuous LODs for rendering. A detailed discussion on the subdivision framework and techniques can be found in [Schröder and Zorin 1998].

Subdivision in 1D is used to create the curves that represent hairs as individual strands, to be discussed in Sec. 3.6. The 4*pt* scheme in 1D is an efficient method for creating a smooth curve. For surface representations, we create a triangular base mesh and subdivide. The control mesh for each of the three representations is shown in Fig. 4(a)(c) and (e).

3.3 The Base Skeleton

Based on the idea for modeling each individual hair strand [Kurihara et al. 1993], we use a similar structure for the base skeleton, which forms the "core" of our proposed set of LOD representations. The base skeleton is comprised of n control points, or nodes. This value is decided automatically based on criteria such as the length of the hair, the waviness or curliness specified for the hair, and the desired smoothness. The skeleton is modeled as an open chain of line segments that connect these nodes. Spring forces are used to control the angles between each node, while the distance between each node is fixed. Fig. 3(a) shows the basic setup of the skeleton. The n nodes $(\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1})$ and $n-1$ segments $(s_1, s_2, \ldots, s_{n-1})$ define the skeleton. Specifying the shape of the skeleton model is discussed in Sec. 3.8.

3.4 Strips

The strip model in Fig. 4(a) and (b) uses a single skeleton model as its basis for motion. The structure for this model is inspired by the strips representation presented by [Koh and Huang 2000; Koh and Huang 2001]. The skeleton is the center of the strip and for each node in the skeleton there are two control points that are used to define the strip. These two strip control points and the skeleton node point are collinear. A skeleton with n nodes will result in a subdivision surface created from a control polygon consisting of $2n$ control points.

A strip is typically used to represent the inner most layers of hair or parts of hair that are *not visible* to the viewer and, therefore, are often not rendered. It is the coarsest (lowest) level-of-detail used for modeling hair. It is mainly used to maintain the global physical behavior and the volume of the hair during the simulation.

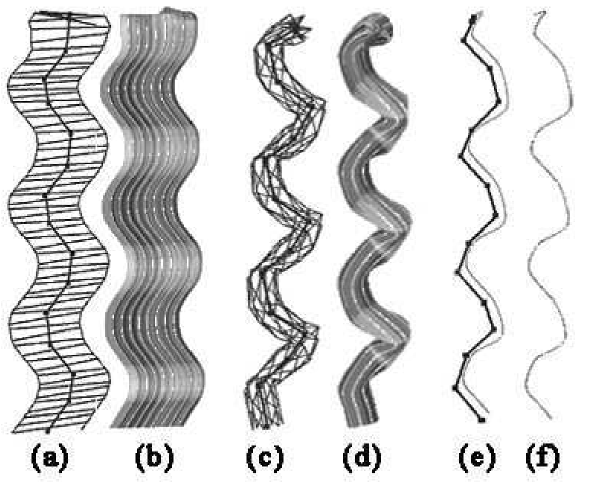

Figure 4: Level-of-Detail Representations for Hair Modeling. *(a) Subdivision representation of strip with skeleton; (b) Rendered strip; (c) Subdivision representation of cluster with skeleton; (d) Rendered cluster; (e) Subdivision representation of a strand with skeleton; (f) Rendered individual strand.*

3.5 Clusters

The clusters are represented as generalized cylinders created with texture-mapped subdivision surfaces, as shown in Fig. 4(c) and (d). Each cluster is based on one skeleton that is located at the center of the cluster. A radius is specified at the top and the bottom of each cluster. The radius is then linearly interpolated at each skeleton node point; this allows the thickness to vary down the length of the cluster. At each skeleton node, a circular cross-section, made up of m control points, is created based on the radius value at that node. Thus, a skeleton made up of n points will create a cluster of mn control points. Typically having m=4 is enough detail to define the cross-section.

A cluster is used to model the intermediate layers of hair and often makes up the majority of the body of semi-visible hair. Whenever appropriate, it is far less costly to represent a group of hair using the cluster model, instead of a large number of individual strands.

3.6 Individual Strands

Each individual strand is modeled as a subdivision curve using 1D subdivision with n control points, as shown in Fig. 4(e) and (f). Most human heads have a few individual strands that are separate from the body of their hair. These types of small imperfections are usually only noticeable when viewing the hair closely. The ability to see these individual strands is what makes this representation a finer detailed model of hair than the clusters.

3.7 Generating LODs

In the model simplification literature, typically either static LODs are generated offline for online switching, or dynamic LODs are computed on the fly. In our current framework, a combination of both static and dynamic LOD representations is used.

Continuous LOD representations are generated by the subdivision of curves and surfaces for fast rendering. Given the three basic discrete LOD representations, a small number of individual strands are grouped into clusters and a few hair clusters are combined to form a hair strip. In our current implementation, for the ease of validating the effectiveness of this framework, a predefined number is used in the simulation. (Due to page limitation, more implementation detail is given at the project website.) However, this can be modified to generate dynamic groupings of strands and clusters on the fly. This is a non-trivial computation to perform in real-time, considering the number of strands or clusters on a human head. It is thus beyond the scope of this paper and we plan to investigate the feasibility of this option in our future work.

We also use a hybrid representation, where only a single skeleton model is used to simulate physical behavior for a group of individual strands, while individual strands are rendered. This is a fairly popular technique used to generate high-quality animation. This is used in some of our simulations to help further accelerate the overall performance, while maintaining the overall visual quality.

3.8 Hairstyling

The skeleton controls the motion and the shape of each hair section and is responsible for the overall style of the hair. Various shapes or styles of hair can be specified by stipulating the rest angles θ_0 and ϕ_0 (see Fig. 3) of each node i of the skeleton. Straight hair can be created by assigning θ_{i0} to 0 and ϕ_{i0} to 0 for each node $\mathbf{p}_i$ of the skeleton. In addition, we can create a wavy hairstyle by zig-zagging the position of the nodes down the length of the skeleton. A zigzag or wavy skeleton is created by assigning each θ_{i0} to a certain angle between 0 and 90 degrees and then the values of ϕ_{i0} alternate by 180 degrees. Ringlet or spiral curls can also be created using the skeleton by specifying an angle value between 0 and 90 degrees for θ_{i0} and then, to achieve the spiral effect, each ϕ_{i0} value increments by 90 degrees down the length of the skeleton.

These two processes for stipulating waves or curls can be altered to create varying styles. The segment size and the values for ϕ_{i0} can be changed to create non-uniform curls and waves according to the desires of the user.

Both the strands and the clusters are able to accurately depict the shape defined by the user. While the strip representation gives better visual results for straight hair, it can also be used to model wavy and curly hair, but not in as fine a detail as the other two representations. Strips are only used when the viewer cannot observe fine detail, such as when the hair is at distances far from the viewer, or when the hair is not in sight. Thus, while the strip cannot depict all hairstyles as accurately as the other two LODs, it is not usually apparent to the viewer. Criteria for choosing an LOD is discussed in further detail in Sec. 7.

4 Dynamic Simulation

The use of the same underlying skeleton model for each hair representation, i.e. strips, clusters, and individual strands, provides a simple yet effective mechanism for switching between different levels-of-detail. In this section, we describe the dynamic model of the skeleton, and explain our hair simulation algorithm that uses a combination of these three representations.

4.1 Basic Physics Model

The physics of motion for the base skeleton is similar to those described in [Anjyo et al. 1992; Kurihara et al. 1993]. The forces that are applied to each node point govern the motion of the skeleton. The force measured from the angular springs of the skeleton model, F_{spring}, helps hold the specified hairstyle during the simulation. $F_{i_{spring}}$, force for the ith node of the skeleton, is calculated by combining the forces for F_{i_θ} and F_{i_ϕ}.

$$F_{i_\theta} = -k_\theta(\theta_i - \theta_{i0}),$$
$$F_{i_\phi} = -k_\phi(\phi_i - \phi_{i0}),$$

where k_θ and k_ϕ are angular spring constants and θ_{i0} and ϕ_{i0} are initial angles for node i.

Other forces that act on the hair are gravity, $F_{gravity}$, external forces, F_{ext}, such as wind, and forces due to collision. We will ignore the influence of collision forces on a node until Sec. 5. Summing the remaining forces together, we obtain a magnitude and direction for the simulation force F_{sim} applied to each skeleton node $\mathbf{p}_i$, to be:

$$F_{i_{sim}} = F_{i_{ext}} + F_{i_{spring}} + F_{i_{gravity}}.$$

4.2 Transitioning between LODs

One of the most crucial aspects of using LODs in a simulation is the ability to switch between different representations smoothly. In order to avoid a sudden jump or popping in the graphical display, it is necessary that the motion and positioning of the hair remain consistent throughout the transition.

Since the motion of each LOD is based on the same underlying skeleton model, we can use this formulation to move from one level to another with little visual artifacts. When a switch is made, the skeleton of the new level-of-detail inherits the dynamics state of the skeleton of the previous level.

For example, if the current LOD is a strip, the algorithm refines the hair model and makes a transition to multiple clusters. The section of hair transitions from a model with one skeleton to one with c skeletons, where c is the number of clusters represented by one strip. Each cluster skeleton linearly interpolates position and motion values from the strip skeleton based on the position of the cluster skeleton's root, or the first skeleton node, $\mathbf{p}_0$, in relation to the root of the strip skeleton.

The transitions are even more straightforward going in the reverse direction. As we move from multiple clusters back to a single strip, the skeleton of the strip simply inherits the average position and motion values of the cluster skeleton that the strip represents. Transitions between the clusters and individual strands are performed in a similar manner.

By using these simplified representations together, our framework automatically switches between different LODs of hair, simplifying the dynamics of the hair as needed. Criterion for switching between LODs is discussed in Sec. 7. A strip can model the largest portion of hair and its simulation requires as little computation as a single strand or a cluster of hair. When appropriate, strips are used to accelerate the simulation of hair, while maintaining some high-level behavior of the hair dynamics. Clusters are used in a similar manner to accelerate the simulation.

5 Collision Detection

Collision detection is a vital part of hair simulation since hair is in constant contact with the scalp of the head and other hairs. Due to the high complexity of hair, this can be a costly computation and it is crucial that the collision detection is performed efficiently.

5.1 Bounding Volume Hierarchy

There are many techniques known for collision detection. Some of the commonly used algorithms for general models are based on the use of bounding volume hierarchies (BVHs). A tree of bounding volumes (BVs) is pre-computed offline to enclose sets of geometric primitives, such as triangles. To perform collision detection using BVHs, two objects are tested by recursively traversing their BVHs [Larsen et al. 2000].

5.2 Swept Sphere Volumes

To perform collision detection on the different representations of hair, we adapt the family of "swept sphere volumes" (SSV) [Larsen et al. 2000] to surround the hair. SSVs are a family of bounding volumes that correspond to a core skeleton grown outward by some offset. The set of core skeletons may include a point, line, or n-gon. We have chosen to use arbitrarily oriented rectangles, instead of n-gons, as the most complex skeleton in our current framework. More precisely, let C be the core skeleton and S be a sphere of radius r. Each SSV, B, can be defined as:

$$B = C \oplus S = \{\boldsymbol{c} + \boldsymbol{r} | \ \boldsymbol{c} \in C, \boldsymbol{r} \in S\}$$

SSVs are chosen as the bounding volumes in our framework, because the shape of our LOD representations shares close resemblance to those of SSVs. Different SSVs provide varying tightness. For clusters and strands, line swept spheres (LSS) are the best candidates for each segment, while rectangle swept spheres provide the better fit for strips, and the point swept sphere for the head at the top level. To detect a collision between a pair of SSVs we simply perform a distance computation on the corresponding pair of core skeletons and then subtract the appropriate offset (radius of each SSV).

5.3 Constructing SSVs for Hair Representations

For each rigid segment of the skeleton model, that is, each line segment between two nodes, we pre-compute an SSV BV. Since each representation of the hair can be tightly encapsulated using a single SSV, a BVH is not necessary for the hair. For a skeleton with n nodes, there are $n - 1$ segments, and thus $n - 1$ single SSVs. The variable thickness of each segment defines the radius of the SSV along its length.

In order to compute a BV for a strip, we let the four control points of the strip that outline a skeletal segment define the area for a BV to enclose. This is performed for each of the $n - 1$ segments along the skeleton. The geometry of the strip is different from the other two representations in that the strip is a surface while the clusters and a collection of strands are volumes. In order to allow the transition from a strip into multiple clusters remain faithful to the volume of hair being depicted we create a BV for a strip section by surrounding it with a box of certain thickness. Each strip is given a thickness equal to that of its cluster and strand grouping counterparts. While the strip is rendered as a surface, it acts physically as a volume. Thus, when a transition from a strip into clusters occurs, the *volume* of hair being represented remains constant throughout this process.

For the cluster representation, we create a BV around the $2m$ control points that define a segment (m control points, as defined in section 3.5, from the cross-section at the top of the segment and m control points at the bottom of the segment).

For individual strands, we perform collision detection for each strand or group of strands, depending on implementation, in a manner similar to that of the clusters. We compute an LSS around the skeleton that defines each segment with a radius defining the thickness. The radius of each BV is varied based on the thickness of a group of strands.

Once the BVs are computed for the hair we construct a BVH of SSVs using a top-town hierarchy construction for the other objects in the scene, head, body, etc., as a pre-computation. During the runtime simulation, the single SSVs and the BVHs are used to perform collision queries, and are lazily updated on the fly. To detect a collision between a segment of hair and an outside object, the hair's SSV is tested against the BVH of the object. (Please refer to [Ward et al. 2003] for more detail.)

5.4 Collision Response

Since the same skeleton model is used for each representation, whenever a collision is detected we calculate the response using the same method for all of the representations. Once a collision between the hair and the head, or other object, has been detected our algorithm determines how much the segment of hair is penetrating the head, the velocity of that segment of hair in the direction of the head is set to zero, and the hair segment is moved so that it is outside of the head. A frictional force in the direction tangent to the head is also applied to each skeleton node when a segment of hair collides with the head or body.

During the simulation, segments of hair are in constant contact with other segments of hair. Since we do not want to perform a collision detection test on a hair segment against all of the remaining segments of hair, the area around the head is spatially decomposed or broken into three-dimensional grids. Each segment of hair is inserted into the grids and only hair segments that fall into the same grid are tested against each other. Segments of hair are tested against each other for collision by performing an intersection test on their SSVs.

If two hair segments intersect with each other, they need to be moved apart. The SSV overlap test determines the distance the skeleton segments are from each other in addition to testing for collisions. We then use this calculated distance as the amount to move the intersecting segments apart.

Further detail on the hair collision detection and collision response algorithms, including exact methods for determining response magnitudes and directions for hair-object and hair-hair collisions and calculating frictional forces, can be found in [Ward et al. 2003].

6 Rendering

In our system, a mixed model of discrete and continuous geometric LODs is used to render the hair representations as described in section 3. We adapted the shading model suggested by [Kajiya and Kay 1989], to capture the anisotropic nature of hair. An efficient implementation of anisotropic surface rendering is performed using an OpenGL texture matrix, as appears in [Heidrich and Seidel 1998]. The rendering of the subdivision surfaces that are used for clusters and strips are assigned two additional textures. The first texture contains the hair color information. It is created as a pre-process from the same color range that assigns color values to the strands. The next texture used on the surfaces contains alpha values that define the transparency of the surfaces [Koh00]. Alpha mapping is used to create the illusion that there are individual strands being rendered. The anisotropic lighting, the hair color, and the alpha textures for the surfaces are rendered in a single pass using multi-texturing.

Self-shadowing is a vital factor to increase the volumetric cue of the hair. We use opacity shadow maps [Kim and Neumann 2001], which are a fast approximation of deep shadow maps.

Aliasing is an innate problem of hair rendering, because each hair strand is too thin to occupy a single pixel. Fortunately it is not as major of a problem for high LOD representations like the clusters and strips, as it is for strands. We rely on graphics hardware for antialiasing. NVIDIA GeForce family provides hardware implemented high resolution antialiasing through multi-sampling [NVIDIA 2002]. We select 4-sample 9-tab multi-sample mode – the highest quality available to us, because various multi-sampling modes provided by the graphics hardware does not affect overall performance of our system. In order to render the simulated hairs with motion blur, we adapt the technique presented by [Haeberli and Akeley 1990]. The current image is rendered into the accumulation buffer so that it can be integrated with previous images. Using the accumulation buffer also reduces the aliasing of individual strands.

Switching between different LODs can introduce visual disturbances. To address this problem, we combine several methods. By using the alpha channel in the textures for clusters and strips we add the visual illusion of strands, making the LOD transition less noticeable. The density values necessary for the opacity shadow map computations for the clusters and strips are based on the number of strands they represent. This helps to keep the brightness of the shadows constant. We also blend the images of previous and current LODs to make transitions smoother.

7 Choosing Hair Representations

Given the three representations for a section of hair, a single representation for modeling and simulating that section is computed on the basis of several criteria. We have used the following components in our current implementation:

- Visibility
- Viewing distance
- Hair motion

7.1 Visibility

If a viewer cannot see a section of hair, that section does not need to be simulated or rendered at its highest resolution. The viewer cannot see hair if it is not in the field of view of the camera or if it is completely occluded by the head or other objects in the scene.

If a section of hair in strand representation is normally simulated using g number of skeletons but is occluded by other objects, we simulate that section of hair using one larger strip, and therefore, one skeleton. When that section of hair comes back into view, it is important that the placement and action of the hair are consistent with the case when no levels-of-detail are used at all, therefore we continue to simulate it. In addition, when a hair section is occluded, it does not need to be rendered at all. Therefore, when a section of hair is occluded, we simulate the hair that might normally be represented as either clusters or strands as strips that use fewer skeletons and these sections are not rendered.

In our current implementation, we perform a simple occlusion test that involves fitting a sphere to the head so that it is slightly smaller than the head. If there are other objects in the scene, such as a body, a similar method is applied. We then test the SSV bounding volumes of the hair to see if they are visible from the camera or if the occluders occlude them. It is possible to use more sophisticated occlusion culling algorithms, such as [Zhang et al. 1997], or special features on new GPUs (e.g. NVidia NV30) to perform these tests more efficiently.

7.2 Viewing Distance

Hair that is far from the viewer cannot be seen in great detail. We can estimate the amount of detail that will be seen by the viewer by computing the screen space area that the hair covers. As the distance from the viewer to the hair increases, the amount of pixels covered by the hair gets smaller and less detail is viewable. We can calculate the amount of pixels covered by the hair to choose the appropriate LOD.

A single strip, a group of c clusters, or a group of g strands represent a given portion of hair. Each is designed to cover a similar amount of world space, thus we can use the control skeleton of the strip as an estimate to the amount of screen space area a given hair section occupies. To determine the screen coverage of this hair, we create a line from the first skeleton node to the last skeleton node and project this line into screen space. If the amount of screen space covered by this line exceeds the pre-determined maximum allowable size for a strip, then the given hair section will be rendered as a cluster. Similarly, if the amount of screen space covered by the line exceeds the maximum allowable size for a cluster, then it will be rendered as individual strands. The pre-determined maximum allowable size for each LOD is decided experimentally based on the viewer's preference.

7.3 Hair Motion

If the hair is not moving at all, then a large amount of computation is not needed to animate it and we can use a lower level-of-detail. When the avatar makes sudden movements, e.g. shaking his or her head, or a large gust of wind blows through the hair, a higher-detailed simulation is used. When a large force is applied to the hair, such as wind, often individual strands can be seen even by a person who is normally too far away to see individual strands of hair that are not in motion.

We choose the particular LOD based on hair motion by first determining the skeleton node in the current representation that has the strongest force acting on it. This value is compared to certain thresholds defined for strands or clusters. If the force acting on the skeleton is not high enough to be represented as either strands or clusters, then the hair can be modeled as a strip.

7.4 Combining Criteria

At any given time during a simulation, a head of hair is represented by multiple LODs. Each section of hair uses its own parameter values to trigger a transition. The sections of hair that have a root location at the top of the head, and therefore more viewable, remain at the individual strands level longer than the sections of hair that are located at the base of the neck. Thus, even if these two sections are at the same distance from the camera and have the same motion, it is more important that the top layer be represented as individual strands instead of clusters, since it is in direct view. When determining an appropriate LOD to use, we first test that section of hair for occlusion. If the hair is not visible to the viewer then we automatically simulate it as a strip and do not render it. In this case, no other transition tests are needed. If the section of hair is visible, we perform the motion and distance tests described above. The LOD representation is chosen based on whichever of these two tests requires higher detail. The use of different representations for the hair is virtually unnoticeable to the viewer.

8 Results and Comparisons

We have implemented our automated simplification algorithm for hair modeling in C++. We modified and extended the publically available proximity query package, PQP [Larsen et al. 2000], to perform collision detection. The simulation results are displayed using OpenGL.

8.1 Implementation Issues

In order to speedup the LOD transitions at runtime, many components are precomputed. An interactive hairstyling tool was created to allow the user to place the skeletons on the head at the desired locations. The styling tool also lets the user set parameters for curly or wavy hair automatically by setting the angles that control the degree of the curl or wave, respectively. The size (length, width, radius, etc.) of each representation is also set using this styling tool.

We also precompute the corresponding BV for each representation of hair to be used for collision detection. Therefore, during an LOD transition, the only values that need to be updated are the positions of the skeleton nodes. Moreover, we use a simplified representation of the head model in performing collision detection between the head and the hair in our simulation.

8.2 Performance Comparisons

We have tested our implementation on various scenarios. Please visit our project website:

http://gamma.cs.unc.edu/HSLOD

for MPEGs of these simulation runs and for a sequence of snapshots taken from a hair simulation using our LOD representations.

We also compared the performance for the overall dynamic simulation (not including collision detection) and collision detection using different representations on various simulation scenarios. Table 1 gives a detailed comparison of the *average* running times using a combination of LOD representations (indicated as LODs) against the use of only one of the three discrete LOD representations (Strands, Clusters, and Strips). Fig. 5 shows the runtime comparison of the simulation performance over the entire duration, as the camera zooms out, increasing the distance to the viewer. The rendering performance is similar to that of the simulation. The basis for our comparisons uses the average timings for the strand simulation as the value 1 on the graph.

For this benchmark, we used 8045 individual strands, which were represented using only 173 strips or only 519 clusters. In these benchmarks, a combination of all three discrete LOD representations was automatically determined by our framework at any given time during the simulations. Timings were taken on a PC equipped with an Intel(R) Pentium(R) 4 2-GHz processor, 1 GB main memory and GeForce(R) 4 graphics card.

Strips provide the best overall performance in simulation time, since it is the coarsest (lowest) LOD of hair. But, a combination of three discrete LOD representations using our framework offers significant performance advantages over the use of individual strands alone. Note there are also implicitly continuous LOD representations used in our system with the subdivision framework. While it renders images of almost equal visual quality as that of individual strands, our LOD implementation gives much better timing performances than modeling with individual strands in simulation, collision detection, as well as rendering.

Breakdown	LODs	Strands	Clusters	Strips
Dyn Sim	0.0175	0.5834	0.0298	0.00592
Col Detect	0.0567	2.1934	0.1297	0.01896
Total	0.0742	2.7768	0.1595	0.02488

Table 1: Performance Comparison. *Simulation for a camera zooming out. The average performance numbers are measured in seconds per frame.*

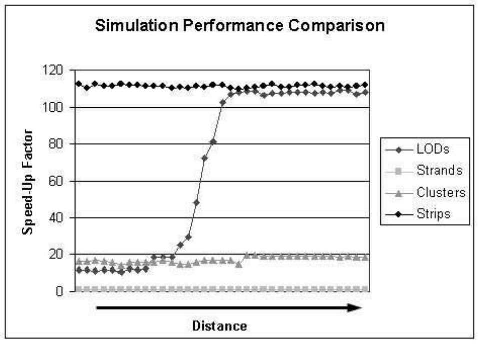

Figure 5: Simulation Performance Comparison. *We show the factors of speed-up among LOD, strips, and clusters over the strands alone, which is the baseline for comparison. The simulation speed of our system consistently outperforms the individual strands. It quickly outperforms the use of clusters alone, as the camera starts to zoom out. Then, soon after a certain distance threshold, it performs comparably to the use of strips alone.*

8.3 Analysis and Discussion

The impetus of this research is to explore the use of level-of-detail representations for modeling hair to automatically generate its *aggregate* behavior, while preserving the visual fidelity of the overall simulation. It is difficult to meaningfully quantify the computational errors introduced by the use of simplified representations for modeling hair. However, we can subjectively evaluate the resulting simulation by performing comparison on the visual quality of rendered images.

Using side-by-side comparison (see the project website), we notice little degradation in the visual quality of the rendered image using LODs. While they offer the best computational performance, the images of hair simulated by strips appear sharp and angular, lacking a realistic appearance. The performance of our framework varies depending on the scenarios. In general, its overall performance in simulation and rendering compares favorably against the use of strands, clusters or strips alone. However, our approach can automatically place the computing resources at places where the hair is most visible to the viewer, and thus offer a much higher visual quality for the resulting simulation.

Limitations: One possible limitation of our algorithm is the slight popping that can occur if aggressive LOD transitions are used. However, we have alleviated this visual artifact with motion blurring and image blending during the LOD transition in our system.

Occlusion culling is an active and challenging area of research. To perform occlusion culling for hair rendering presents many more new challenges, as the hair can self occlude, but each strand, cluster or strip is rather small in size yet in aggregation its capability to occlude the rest of the hair can be significant. Our current implementation only has simple object-hair occlusion tests to validate the basic idea. However, we have already observed some performance gain using occlusion culling. We plan to investigate more sophisticated techniques, including the use of new graphics hardware, in the near future.

One of the foremost difficulties in hair simulation is to achieve stability for all types of hairstyles, lengths and interactions. We have implemented Runge-Kutter of order 4 for numerical integration. We are currently investigating the use of implicit methods.

In order to achieve high rendering rates, we are only performing two-pass refinement in our current implementation of opacity shadow maps. For better rendered images, more passes are required

[Kim and Neumann 2001]. We plan to develop a nicer interface to allow the users to trade off speed for higher-quality rendering and vice versa.

There are other application dependent transition criteria, such as collision, that we have not examined closely. These factors can further contribute to improving the overall system performance.

8.4 Comparisons Against Other Approaches

A multiresolution technique for hairstyling is presented in [Kim and Neumann 2002]. They only use groups of strands of different sizes. They have not applied this approach to hair simulation. The switching is directly controlled by the user, while ours offers the capability of automatic switching. To apply this technique to hair simulation requires dynamic grouping on the fly. This is very difficult to perform given the number of strands.

Techniques that use interpolation from guide strands, such as [Chang et al. 2002], can be limiting in the types of hairstyles that can be modeled as well as interaction capabilities. Furthermore, their simulations run at slower rates than ours [Yu 2002].

Our approach compares favorably to those using only cluster-like representations (e.g. wisps or generalized cylinders) [Chen et al. 1999; Kurihara et al. 1993; Xu and Yang 2001; Plante et al. 2001], as we can achieve similar visual quality with faster rendering rates.

The real-time animation techniques [Koh and Huang 2000; Koh and Huang 2001; Lengyel 2000; Lengyel et al. 2001; NVIDIA 2001] and those commonly found in video games are either limited to certain hairstyles (mostly short, straight hair) or the resulting image quality is inferior to ours.

9 Summary and Future Work

In this paper, we present the use of levels-of-detail for modeling hair to accelerate dynamics computation, simplify collision detection and reduce rendering costs. In addition to potential areas of improvements mentioned in the earlier sections, there are several possible directions to extend this research:

- Interactively modify the dynamics of the hair in the presence of other substances, such as water, styling gel, hair spray, etc.;
- Dynamically change the hairstyle, as the user combs or brushes the hair with a 3D user interface;
- Automatically generate desired simulation outcomes, given high-level user guidance.

Acknowledgements

This research is supported in part by Army Research Office, Intel Corporation, National Science Foundation, and Office of Naval Research. We would like to thank Ben Lok for the body model and help in capturing head motion.

References

ANJYO, K., USAMI, Y., AND KURIHARA, T. 1992. A simple method for extracting the natural beauty of hair. *Computer Graphics 26*, 2, 111–120.

CARLSON, D., AND HODGINS, J. 1997. Simulation levels of detail for real-time animation. In *Proc. of Graphics Interface 1997*.

CHANG, J., JIN, J., AND YU, Y. 2002. A practical model for hair mutual interactions. *Proc. of ACM Symposium on Computer Animation*.

CHEN, L. H., SAEYOR, S., DOHI, H., AND ISHIZUKA, M. 1999. A system of 3d hair style synthesis based on the wisp model. *Visual Computer 15*, 4, 159–170.

CHENNEY, S., AND FORSYTH, D. 1997. View-dependent culling of dynamic systems in virtual environments. In *Proc. of ACM Symposium on Interactive 3D Graphics*.

DALDEGAN, A., KURIHARA, T., MAGNENAT-THALMANN, N., AND THALMANN, D. 1993. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum (Proc. of Eurographics 12*, 3, 211–221.

FUNKHOUSER, T. A., AND SÉQUIN, C. H. 1993. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proc. of ACM SIGGRAPH*, J. T. Kajiya, Ed., 247–254.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proc. of ACM SIGGRAPH*, 9–20.

HADAP, S., AND MAGNENAT-THALMANN, N. 2001. Modeling dynamic hair as a continuum. *Computer Graphics Forum (Proc. of Eurographics 2001) 20*, 3.

HAEBERLI, P. E., AND AKELEY, K. 1990. The accumulation buffer: Hardware support for high-quality rendering. In *Proc. of ACM SIGGRAPH*, F. Baskett, Ed., 309–318.

HEIDRICH, W., AND SEIDEL, H.-P. 1998. Efficient rendering of anisotropic surfaces using computer graphics hardware. *Proc. of Image and Multi-dimensional Digital Signal Processing Workshop (IMDSP)*.

KAJIYA, J. T., AND KAY, T. L. 1989. Rendering fur with three dimensional textures. In *Proc. of ACM SIGGRAPH*, J. Lane, Ed., 271–280.

KIM, T.-Y., AND NEUMANN, U. 2000. A thin shell volume for modeling human hair. *Computer Animation*.

KIM, T.-Y., AND NEUMANN, U. 2001. Opacity shadow maps. *Proc. of Eurographics Rendering Workshop*.

KIM, T.-Y., AND NEUMANN, U. 2002. Interactive multiresolution hair modeling and editing. In *Proc. of ACM SIGGRAPH*. 620–629

KOH, C. K., AND HUANG, Z. 2000. Real-time animation of human hair modeled in strip. *Eurographics CAS Workshop*, 101–112.

KOH, C. K., AND HUANG, Z. 2001. A simple physics model to animate human hair modeled in 2d strips in real time. *Proc. of Eurographics Workshop on Animation and Simulation*.

KURIHARA, T., ANJYO, K., AND THALMANN, D. 1993. Hair animation with collision detection. In *Models and Techniques in Computer Animation*, Springer-Verlag, 128–38.

LARSEN, E., GOTTSCHALK, S., LIN, M., AND MANOCHA, D. 2000. Distance queries with rectangular swept sphere volumes. *Proc. of IEEE Int. Conference on Robotics and Automation*.

LEBLANC, A. M., TURNER, R., AND THALMANN, D. 1991. Rendering hair using pixel blending and shadow buffers. *The Journal of Visualization and Computer Animation*.

LENGYEL, J. 2000. Real-time fur. *Proc. of Eurogrpahics Workshop on Rendering*.

LENGYEL, J., PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2001. Real-time fur over arbitrary surfaces. *Proc. of ACM Symp. on Interactive 3D Graphics*.

LUEBKE, D. 2001. A developer's survey of polygon simplification algorithms. *IEEE CG & A* (May), 24–35.

MAGNENAT-THALMANN, N., HADAP, S., AND KALRA, P. 2000. State of the art in hair simulation. *Int. Workshop on Human Modeling and Animaton*, 3–9.

NVIDIA. 2001. Final fantasy technology demo 2001. *http://www.nvidia.com*.

NVIDIA. 2002. *http://developer.nvidia.com/docs/lo/1451/SUPP/accuview.final.pdf* .

O'BRIEN, D., FISHER, S., AND LIN, M. 2001. Simulation level of detail for automatic simplification of particle system dynamics. *Proc. of Computer Animation*, 210–219.

PERBET, F., AND CANI, M. 2001. Animating prairies in real-time. *Proc. of ACM Symposium on Interactive 3D graphics*.

PLANTE, E., CANI, M., AND POULIN, P. 2001. A layered wisp model for simulating interactions inside long hair. *Proc. of Eurographics Workshop on Animation and Simulation*.

SCHRÖDER, P., AND ZORIN, D. 1998. Subdivision for modeling and animation. In *ACM SIGGRAPH Course Notes*.

WARD, K., LIN, M. C. AND MACRI, D. Collision Detection for Animating Hair Using Multiresolution Respresentations. Technical Report, University of North Carolina at Chapel Hill. January 2003.

XU, Z., AND YANG, X. D. 2001. V-hairstudio: An interactive tool for hair design. *IEEE Computer Graphics and Applications 21*, 3, 36 –43.

YU, Y. 2001. Modeling realistic virtual hairstyles. *Pacific Graphics*.

YU, Y. 2002. Personal Communication.

ZHANG, H., MANOCHA, D., HUDSON, T., AND HOFF, K. 1997. Visibility culling using hierarchical occlusion maps. In *Proc. of ACM SIGGRAPH*, 77–88