Adaptive Grouping and Subdivision for Simulating Hair Dynamics

Kelly Ward Ming C. Lin
Department of Computer Science
University of North Carolina at Chapel Hill
{wardk,lin}@cs.unc.edu
http://gamma.cs.unc.edu/HAIR

Abstract

We present a novel approach for adaptively grouping and subdividing hair using discrete level-of-detail (LOD) representations. The set of discrete LODs include hair strands, clusters and strips. Their dynamic behavior is controlled by a base skeleton. The base skeletons are subdivided and grouped to form clustering hierarchies using a quad-tree data structure during the precomputation. At run time, our algorithm traverses the hierarchy to create continuous LODs on the fly and chooses both the appropriate discrete and continuous hair LOD representations based on the motion, the visibility, and the viewing distance of the hair from the viewer. Our collision detection for hair represented by the proposed LODs relies on a family of "swept sphere volumes" for fast and accurate intersection computations. We also use an implicit integration method to achieve simulation stability while allowing us to take large time steps. Together, these approaches for hair simulation and collision detection offer the flexibility to balance between the overall performance and visual quality of the animated hair. Furthermore, our approach is capable of modeling various styles, lengths, and motion of hair.

1 Introduction

Modeling and animating human characters present some of the most difficult problems in computer graphics. Simulating natural hair movement is crucial to generating realistic appearances of animated characters. Real-time applications, such as virtual environments and game development, pose additional computational challenges for modeling the dynamics and mutual interactions of human hair at interactive rates. Such challenges primarily stem from the high number of hair strands required to model a human character and the resulting complexity of mutual interactions among many nonrigid hair strands. In addition to modeling virtual humans, the techniques used to simulate complex interactions among many deformable strands can also be applied to modeling the motion of long animal hair and manes, brushes, bristles, strings, and other synthetic fibers.

Figure 1. Dynamic Simulation of Long Hair Using Adaptive Grouping and Subdivision of Hair Strands, Hair Clusters, and Hair Strips.

Main Contribution: Herein we present a novel approach to adaptively subdivide and group hair modeled using discrete and continuous level-of-detail (LOD) representations. The set of discrete LOD representations include hair strands, clusters, and strips represented by subdivision curves, subdivision swept volumes, and subdivision patches, respectively [24]. Each representation has a base skeleton used to simulate the hair dynamics, which aids in switching between different LOD representations during the simulation [24]. To accelerate the simulation performance even more while achieving higher visual quality, we precompute a clustering hierarchy using the quad-tree data structure, which generates continuous LODs on the fly, based on the root location of each skeleton. At runtime, our algorithm selects both the appropriate continuous and discrete LOD representations based on the viewing distance, the motion and the visibility of the hair. Complementing the continuous LOD representations generated by the dynamic grouping and subdivision of hair, we use a collision detection algorithm based on a family of swept sphere volumes and an implicit integration method to achieve greater stability while allowing the simulation to take larger time steps. The resulting method has the following characteristics:

- It can smoothly and dynamically switch between any hair LOD representations, as opposed to transitioning between discrete hair representations [24].
- It can incorporate any switching criteria (e.g. the viewing distance, the visibility, the motion of hair) to automatically group and subdivide hair representations and seamlessly control continuous LOD generation.
- It can balance between the visual quality and the performance of the overall dynamic simulation.
- It can be applied to simulate mutual hair interactions for various hairstyles of different thickness, weight, and strength.

We demonstrate our algorithm on several simulation scenarios, including hair braiding, hair shaking abruptly, hair brushing against a complex wrinkled torus, and hair blowing in the wind. We observed noticeable performance improvement with higher visual quality, as compared to earlier techniques, such as [24].

Organization: The rest of the paper is organized as follows. Sec. 2 sets forth a synopsis of related work. Sec. 3 describes the three discrete representations of hair upon which this work is based. We present our method on constructing clustering hierarchies for hair strands, hair clusters, and hair patches in Sec. 4. The collision detection method and the dynamics model for simulating hair movement and interactions are described in Sec. 5. The criteria for performing automatic subdivision and grouping of hair representations on the fly are outlined in Sec. 6. Sec. 7 describes the results of our prototype implementation and compares its performance against earlier techniques.

2 Related Work

In this section, we briefly survey prior research in hair modeling and simulation levels of detail.

2.1 Hair Modeling

Modeling hair has been an active area of research in computer graphics and numerous approaches have been proposed to address the problem it presents [7, 11, 19, 26]. Some fundamental techniques have been presented to model the motion of individual hair strands in [1, 6, 14] in which each hair strand was represented as a series of connected line segments and the shape of the hair was determined by specifying the desired angles between segments. 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* [14]. 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 [23]. A similar approach has been used for interactive hairstyling [5, 25]. Another approach, as evidenced in [4], has been to use adaptive guide hairs to add more detail to overly interpolated regions.

None of these techniques however, can perform hair animation or rendering in real-time. Recently, a thin shell volume [9] and 2D strips [12, 13] 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 [16, 17, 20]. The foregoing techniques, however, do not work well or are not applicable for rendering long, wavy or curly hair.

2.2 Simulation Levels of Detail

Model simplification has been an active research area and many algorithms have been proposed to accelerate graphical rendering of complex environments. Similar approaches have also been suggested to accelerate dynamic simulations for complex systems. A survey on geometric and simulation levels of detail can be found in [18] and [24], respectively. Our work bears some resemblance to the approach proposed by [21] for simplifying dynamics of particle systems using clustering. However, our approach for generating continuous LODs of hair modeled using a combination of hair strands, clusters and strips is quite different and the switching between different LODs is far more complex. Recently, the work of [3] has used the notion of continuous LODs for hair simulation in order to achieve faster simulation results. While our work employs similar techniques as [3], we use different splitting and grouping methods and couple the continuous LODs with discrete LODs to acheive faster rendering and simulation results.

3 Preliminaries

Our approach is built upon the use of three discrete LOD representations for hair that utilize the subdivision framework and a base skeleton [24]. The three LOD representations are texture-mapped individual strands, clusters and strips. They are represented by subdivision curves, subdivision swept volume and subdivision patches, respectively. They are tessellated on the fly to further generate adaptive, continuous LODs for rendering only. Fig. 2 shows the rendered images of each LOD representation.

We model each skeleton as a series of rigid line segments connected by node points. Fig. 3 shows the layout of a skeleton. A skeleton contains n node points,

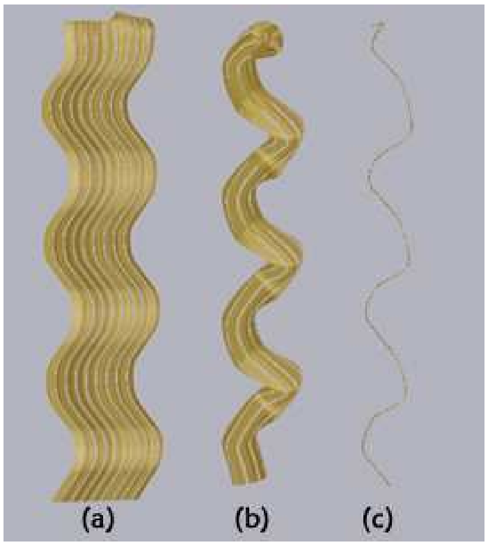

Figure 2. Three Discrete LODs for Hair. *(a) Hair Strip (b) Hair Cluster (c) Hair Strand.*

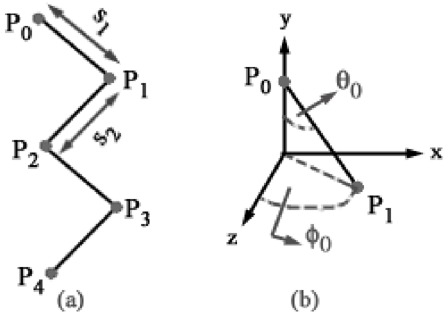

Figure 3. *(a) The base skeleton model; (b) The parameters that define the style of hair.*

$(p_0, p_1, \ldots, p_{n-1})$ and $n-1$ rigid line segments between the points $(s_1, s_2, \ldots, s_{n-1})$. Springs are used to control the angles between each line segment. The resting style of a skeleton is specified by assigning the desired rest angle positions to θ_0 and ϕ_0. In this paper, we have extended the foregoing approach [24] by adaptively subdividing and grouping hair strands, clusters and strips, to generate continuous LODs for simulating complex hair motion and mutual interactions.

4 Construction of Hair Hierarchy

Our LOD algorithm uses the continual subdivision of strand groups, clusters, and strips in order to attain varying detail based on the current simulation state. The subdivision is performed as a pre-process and the information is stored for retrieval during runtime. As the subdivision of a hair group is performed, more skeletons are added to the system, creating a more detailed simulation. Our strand group hierarchy is built in a top-down manner creating smaller groups of strands until we reach the limit of a single strand in a group. Likewise, the hierarchies of clusters are built from top to bottom, as are the strip hierarchies.

We couple the strand group hierarchies with the cluster and strip hierarchies to attain both discrete and continuous LOD representations. This hair representation hierarchy is illustrated in Fig. 4. Assumed to be given at the initial setup, the root strip in the strip hierarchy is the coarsest representation for an assemblage of hair. In order to gain more detail we traverse down the strip tree until we reach its leaves. For more detail, a finest LOD strip representation transforms into the coarsest LOD cluster representation, or the root cluster in the cluster hierarchy. Similarly, to attain even more detail we traverse down the cluster hierarchy until we reach the leaves of the cluster tree. At this point, the cluster representations are replaced by the top-level strand groups of the strand group hierarchies. To attain the finest detailed simulation possible, we traverse to the leaves of the strand group trees, which contain individual strands.

The next sections explain our subdivision process and hierarchy building mechanisms starting with the creation of strip and cluster hierarchies.

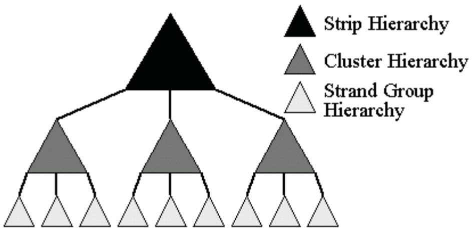

Figure 4. Hair Hierarchy. *One hair hierarchy consists of a single strip hierarchy, multiple cluster hierarchies and multiple strand group hierarchies. The coarsest hair representations are located in the strip hierarchy at the top of the overall hair hierarchy. Note: The number of hierarchies, or children per node, within a hair hierarchy fluctuates.*

4.1 Strip and Cluster Subdivision

Before we can build a hierarchy of strips or clusters, we must first create the initial top-level strip. A top-level strip is created by choosing a location on the scalp for the origin of the skeleton (the first node point of the skeleton). Next, a user-defined width is specified controlling the thickness of the strip. Hairstyle specific information is then declared, defining the length of the hair, the number of control points of the skeleton, and the desired curls or waves down the length of the skeleton.

Because the strip is a two-dimensional surface, we restrict its subdivision such that it may only be split into two equal parts. Strip subdivision is simply the degenerate case to cluster or strand group subdivision, using a degenerate quad-tree, or a binary tree, instead of the quad-tree data structure that is used for cluster and strand group hierar-

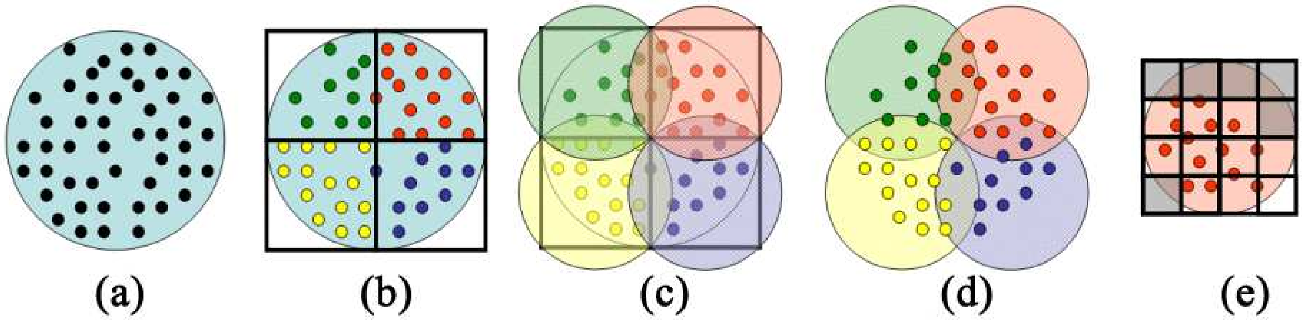

Figure 5. Strand group subdivision. *The subdivision process of a strand group into multiple strand groups. (a) The cross-section of a single strand group. (b) Strand group is divided into 4 equal quadrants and the strands are separated by the quadrant in which they lie (designated by different shades). (c) Circular cross-section is fit around each quadrant, or child, of original strand grouping. (d) Four new strand groups are created which are children of the original strand group. (e) Continual subdivision process is repeated on each child. Tinted squares show empty quadrants that contain no strands, these quadrants are set to null.*

chies. The subdivision ends once the width of the current strip is below a user-defined threshold.

For cluster subdivision, we start with a circular cross-section that defines the cluster. This circular cross-section is then split into four equal parts. The four sub-clusters have the same radius value but represent four different quadrants of the original cluster. The subdivision of a cluster always results in four children, so its information is held in a quad-tree. Clusters stop subdividing once their radius is below a user-defined threshold value. At this point, further detail is created in the strand group hierarchies.

4.2 Strand Group Subdivision

A strand group is defined by a single skeleton, a radius to define the circular cross-section of the group, and the individual strands of hair for rendering purposes. A strand group cross-section is illustrated in Fig. 5a. The individual hair strands are randomly placed within the group and follow the dynamics of the skeleton. The circular shape of the strand groups is used for its simplicity in collision detection, explained in Sec. 5.

We use a quad-tree data structure to contain the hierarchy information. It follows therefore, that each strand group is split into four equal sections, as shown in Fig. 5b. The subdivision of a strand group into four sections creates the tightest fitting circular cross-section possible for each sub-group, as in Fig. 5c and Fig. 5d.

Once we have divided the strand group, we then calculate the number of strands in each quadrant. If a quadrant has no strands within its boundaries then the child associated with that quadrant is set to null (see Fig. 5e). A strand group will have between zero and four children. A strand group that contains only one strand will have zero children and becomes a leaf in the tree.

The final strand hierarchy is depicted in Fig. 6. Each node in the hierarchy contains a strand group, which includes its skeleton and the hair geometry used for the final rendering stage. In addition, each strand group, as well as each cluster and strip, holds its $n-1$ bounding volumes used for collision detection, where n is the number of nodes in the skeleton. The creation of these bounding volumes is described in Sec. 5.

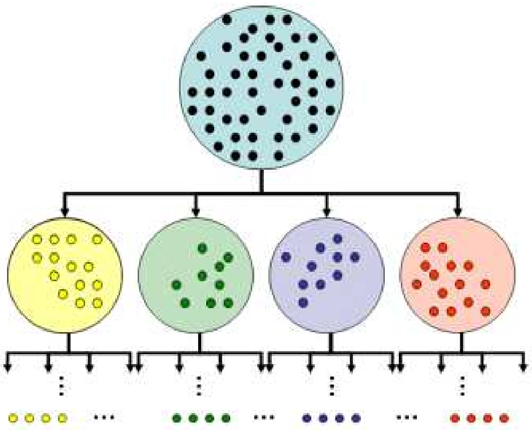

Figure 6. Strand group hierarchy. *Subdivision process creates a quad-tree containing strand group information. Strand group hierarchy can extend to individual strands.*

Each skeleton contains the same number of control points as its parent hair group, which aids in dynamically switching between different levels, described in Sec. 6.

5 Collision Detection and Response

Collision detection and response is usually the most time consuming process for the overall simulation. We have separated our collision algorithm into two parts: *object-hair* collision detection, which occurs when hair interacts with an outside object like the head, and *hair-hair* collision detection, which involves hair mutual interactions.

5.1 Swept Sphere Volumes

Many techniques have been introduced for collision detection. Common practices have used bounding volumes (BVs) as a method to encapsulate a complex object within a simpler approximation of said object.

We have chosen the family of "swept sphere volumes" (SSVs) [15] to surround the hair. SSVs comprise a family of bounding volumes defined by a core skeleton grown outward by some offset. The set of core skeletons may include a point, line, or ngon. Fig. 7 shows examples of some SSVs. To calculate an SSV, let C denote the core skeleton and S be a sphere of radius r, the resulting SSV is defined as:

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

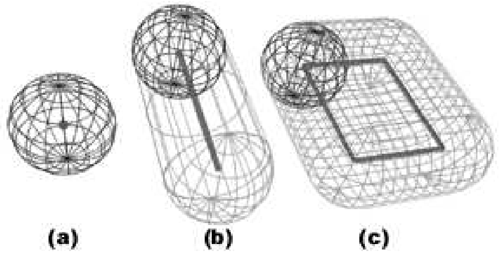

Figure 7. A Family of Swept Sphere Volumes. *(a) Point swept sphere (PSS); (b) Line swept sphere (LSS); (c) Rectangle swept sphere (RSS). The core skeleton is shown as a bold line or point.*

To detect an intersection between a pair of arbitrary SSVs we simply perform a distance test between their corresponding core skeletons and then subtract the appropriate offsets, i.e. the radius of each SSV.

5.2 Swept Sphere Volumes for Hair

We utilize the family of SSVs to encapsulate the hair because the shape of the SSVs closely matches the geometry of our hair representations. The SSVs that correspond to our three geometric representations for hair are line swept spheres (LSSs) for the strands and cluster levels, and rectangular swept spheres (RSSs) for the strip level. These SSVs can be used in combination to detect collisions between different representations of hair.

A skeleton containing n control points will contain $n - 1$ SSVs. These SSVs correspond to the $n - 1$ rigid line segments contained in the skeleton, see Fig. 3. The rigid line segment is used as the core skeleton of the LSS and the radius of the SSV is determined from the thickness of the hair representation.

5.3 Hair-Hair Interactions

Because hair is in constant contact with surrounding hair, interactions among hair are important to capture. The typical human head has thousands of hairs. Consequently, testing the $n-1$ sections of each hair group against the remaining sections of hair would be too overwhelming for the simulation. Instead, we spatially decompose the area around the hair into three-dimensional grids and insert each SSV of the hair into the grids. Only SSVs that fall into the same grid are tested against each other. The average length of the rigid line segments of the skeletons is used as the height, width, and depth of each grid cell.

For each pair of SSVs that falls into the same grid cell, we determine the distance between their corresponding core skeletons, $s1$ and $s2$. This distance, d, is subtracted from the sum of the radii of the two SSVs, $r1$ and $r2$, to determine if there is an intersection. Let

$$overlap = d - (r1 + r2)$$

If $overlap$ is positive then the sections of hair do not overlap and no response is calculated.

If there is an intersection, then we compute the cross product between the core skeletons, $s1$ and $s2$, to determine the orientation of the skeletons in relation to each other. If $s1$ and $s2$ are near parallel, we set their corresponding velocities to the average of their initial velocities.

Intersecting hair sections that are not of similar orientation are pushed apart based on their amount of overlap. The direction to move each hair section is determined by calculating a vector from the closest point on $s1$ to the closest point on $s2$. Each section is moved by half the overlap value and in opposite directions along the vector from $s1$ to $s2$. Fig. 8 shows the effects of hair-hair interactions.

5.4 Hair-Object Interactions

Hair can interact with any object in the scene, such as the head or body of the character, where the object is a solid body that allows no penetration. Our hair-object collision detection algorithm begins by encapsulating the object with a bounding volume hierarchy (BVH) of SSVs that is precomputed offline. A collision is detected between a section of hair and the object by recursively traversing the BVH testing the hair's SSV against the bounding volumes in the hierarchy. If the hair is colliding with the object, the BVH will return the triangles in direct contact with the hair.

If a section of hair is colliding with the object, we adjust the position of the hair section so that it is outside of the object. We determine the amount by which to push the hair section by calculating the amount of penetration of the hair section into the object. We then push the skeleton in the direction normal to the object in the amount of the penetration. The section of hair is now no longer colliding with the object. In addition, the velocity of the section of hair interacting with the object is set to zero so that the hair is restricted to only move tangential to and away from, the object.

In the next time step, we know that the hair is still in

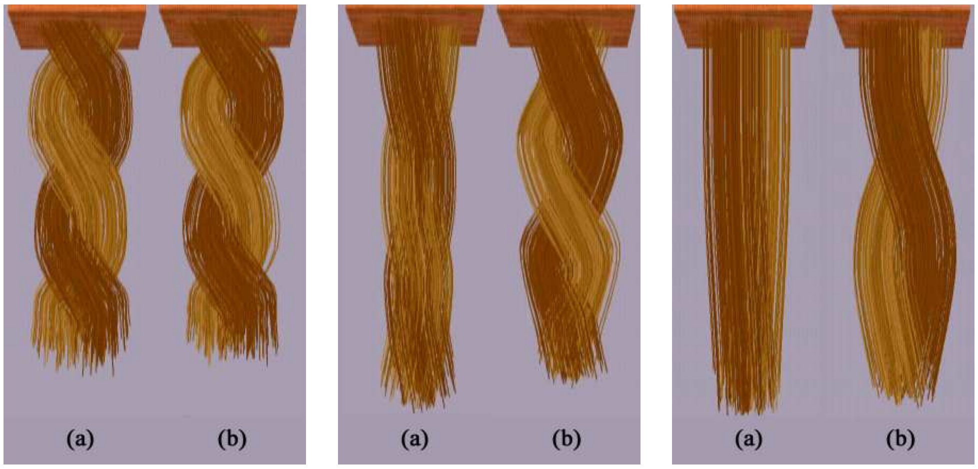

Figure 8. Effects of Hair-Hair Collision Detection. *Side-by-side comparison with (RIGHT) and without (LEFT) hair-hair collision detection in a sequence of simulation snapshots.*

close proximity to the object. If there is no intersection between the object and the hair we determine whether the hair is still within a certain distance threshold. If it is within this threshold, then the hair is still restricted so that its velocity in the direction of the object is zero. If it is not within this threshold, then the hair can move about freely.

When hair interacts with an object, a frictional force must be applied. We calculate a friction force by projecting the acceleration of the hair onto the plane tangential to the object at the point of contact. The result is the acceleration component that is tangent to the object. We apply the friction force in the opposite direction to oppose the motion of the hair. The magnitude of this force is based on the acceleration of the hair and the frictional coefficient, μ_f, which is dependent upon the surface of the object, where $0 < \mu_f < 1$.

5.5 Overall Collision Checking Algorithm

During a single time step, a section of hair can have many interactions, some with other hairs and some with outside objects. The order in which we process these interactions is important as certain interactions must allow no penetration. Therefore, we process the hair-hair interactions first. The avoidance of hair-hair intersections is a soft constraint. Where possible, hair-hair intersections will be avoided, especially in the case of intersecting hair sections of different orientations. In contrast, the avoidance of hair-object intersections is a hard constraint. At the end of the time step, there will be no intersections between the hair and an outside object.

5.6 Implicit Integration for Dynamic Simulation

After our system computes appropriate collision responses, the dynamic simulation proceeds. We follow the basic dynamics model for simulating hair that was first proposed by [1, 14]. We extend this method by using an implicit integration technique, in order to achieve greater stability while allowing us to take larger time steps throughout the simulation. This approach is similar to cloth simulations that use implicit integration for great stability [2].

In this approach, each control point of a hair skeleton is governed by the set of ordinary differential equations:

$$I_i \frac{d^2\theta_i}{dt^2} + \gamma_i \frac{d\theta_i}{dt} = M_{\theta i},$$
$$I_i \frac{d^2\phi_i}{dt^2} + \gamma_i \frac{d\phi_i}{dt} = M_{\phi i}.$$

where I_i is the moment of inertia for the ith control point of the skeleton, γ_i is the damping coefficient, and M_θ and M_ϕ are the θ and ϕ torque components, respectively. M_θ and M_ϕ are computed from the spring forces controlling the style of the hair and external forces such as wind. The resultant M_θ and M_ϕ become:

$$M_\theta = M_{\theta_{spring}} + M_{\theta_{external}},$$
$$M_\phi = M_{\phi_{spring}} + M_{\phi_{external}}.$$

The torques due to the spring forces are calculated by:

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

where k_θ and k_ϕ are the spring constants for θ and ϕ, respectively. Furthermore, θ_0 and ϕ_0 are the specified rest angles and θ and ϕ are the current angle values. Although explicit methods such as Euler or fourth-order Runge-Kutter can be used for this integration, we choose implicit integration for greater stability of simulations. Appendix A shows the derivation of our implicit integration equations using polar coordinates. Because we are working with polar coordinates, we will use angular positions, θ and ϕ, and angular velocities, ω_θ and ω_ϕ.

The change in angular velocity for the θ-component of a skeleton node point, $\triangle\omega_\theta$, becomes

$$\triangle\omega_\theta = \frac{-hk_\theta(\theta - \theta_0) - h^2 k_\theta \omega_{\theta 0}}{1 + h^2 k_\theta}$$

where h is the time step, and $\omega_{\theta 0} = \omega_\theta(t_0)$ is the angular velocity at time t_0. Here, $\triangle\omega_\theta = \omega_\theta(t_0+h) - \omega_\theta(t_0)$. Once we have calculated $\triangle\omega_\theta$, we calculate the change in angular position $\triangle\theta$ from $\triangle\theta = h(\omega_0 + \triangle\omega)$. The same process can be applied to the ϕ-component of the angular position and angular velocity for each control point of a skeleton.

Implicit integration allows us to use stiffer springs when warranted, for example, when simulating the bristles of a brush which have different spring constants than the hair on a human head. Using stiff springs with explicit integration on the other hand, requires much smaller time steps to ensure a stable simulation.

6 Runtime Selection

Our hair hierarchies allow us to choose appropriate discrete and continuous LOD representations for the hair dynamically during the simulation. We simply traverse the hierarchy selecting the desired hair assemblage. As we move to a different level in the hair hierarchy we are either dividing a hair group into multiple groups or combining several groups into one larger group of hair. The base skeleton makes these transitions smooth and straightforward. Because each hair representation uses the same dynamic skeleton, we generalize the transitioning algorithm so that it can be applied at any location in the hierarchy.

6.1 Criteria for Grouping and Subdividing

The current assemblage of hair is determined by the viewing distance from the viewer to the hair, the motion of the hair, and the visibility of the hair. We use a technique similar to that described by [24]. This method first tests the hair's visibility by determining if it is outside of the field of view of the camera or if it is occluded by the body. If a section of hair is not visible, it is simulated using the coarsest representation, the root strip of the hierarchy, and it is not rendered.

The next criterion for choosing the hair's representation is the viewing distance. As the distance from the viewer to the hair increases we move up the hair hierarchy, simulating and rendering the hair with less detail. Meanwhile, as the velocity of the hair increases, we move down the hierarchy, simulating and rendering the hair with finer detail. Each level in the hierarchy has predetermined intervals for its appropriate viewing distances and velocities. These predetermined values are set by defining the intervals for the coarsest LOD in the hierarchy and for the finest LOD in the hierarchy. The remaining values for the rest of the levels are linearly interpolated from the start and end values to create a smooth progression of distance and velocity thresholds.

If a section of hair is not occluded, then its distance and velocity are compared against the current level's thresholds. The current level is chosen based on whichever test requires more detail. In a given time step, a section of hair only moves one level in the hierarchy in order to avoid visual distractions, unless a transition is triggered by occlusion.

A transition caused by an occlusion permits the hair representation to transform into the coarsest strip representation regardless of its current location in the hair hierarchy. When the hair is no longer occluded, it transforms into the LOD that is appropriate given the hair's current velocity and distance values.

6.2 Adaptive Subdivision

Using our precomputed hierarchy, we divide a group of hair into multiple groups by moving a level down the hierarchy. This becomes a simple process through the use of the base skeleton. As explained in Sec. 4, each hair group's skeleton has the same number of control points as its parent skeleton. Furthermore, all of the style properties are the same from parent to child. Accordingly, when a transition to a hair group's children occurs, the child skeletons inherit the dynamic state of their parent skeleton. Each control point in a child skeleton corresponds to a control point in its parent skeleton. When the child groups are created from the parent group, the offset of each child from the parent is stored. When we switch to the children, these offsets are used to position the children accordingly.

Fig. 9 shows two skeletons dynamically subdivide into multiple skeletons as a gust of wind blows through the hair.

6.3 Adaptive Merging

Merging multiple child skeletons back into their parent skeleton is, again, rather straightforward. Our method averages the dynamic states of the children, including position and velocity values, and assigns this average to the parent skeleton.

In order to alleviate visual artifacts that can appear by merging children into a parent skeleton, a transition may only occur if all of the children are ready to transition back into the parent; that is, the criteria explained in Sec. 6.1 for switching levels are satisfied for all of the children. Furthermore, in order to avoid a sudden jump in the position of the hair, we impose a positional constraint on the children. After we have averaged the control point positions of the child skeletons, we determine the distance of the child control points from their corresponding parent control point. If this distance is greater than a certain threshold, the transition will not occur. If the distance is less than the threshold but not in exact position, a spring force is used to subtly pull the children into place so a smooth transition may occur.

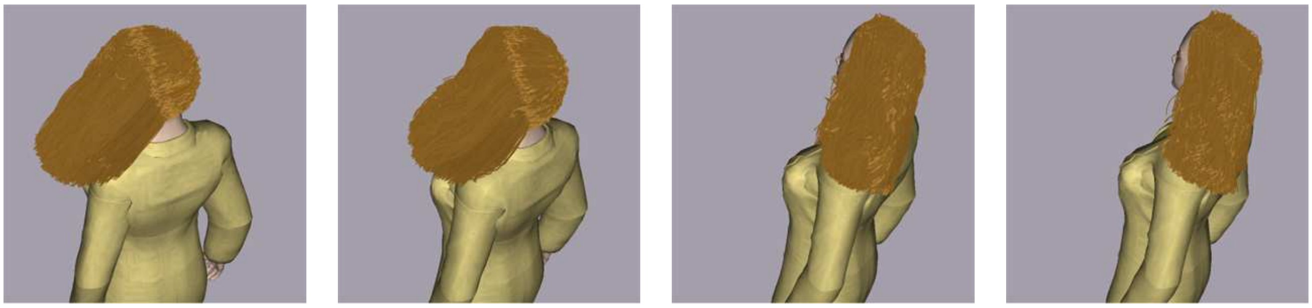

Figure 10. Dynamic Simulation of Hair Using LODs. *A sequence of snapshots (from left to right).*

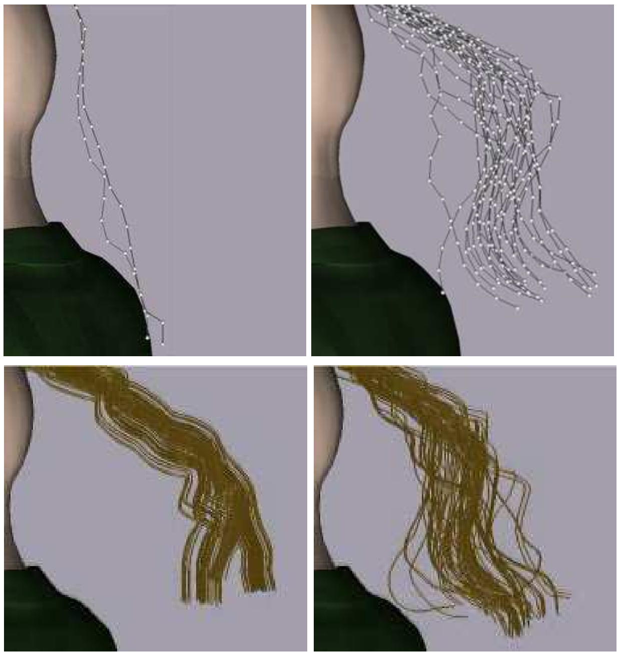

Figure 9. TOP: Two skeletons (LEFT) are dynamically subdivided into multiple (RIGHT). BOTTOM: The rendered images without (LEFT) and with (RIGHT) adaptive subdivision.

7 Results and Comparisons

We have implemented our LOD hair simulation algorithm in C++. We modified and extended the publicly available proximity query package (PQP) [15], to perform collision detections. The simulation results are displayed using OpenGL.

7.1 Rendering

Our rendering system follows that described in [24] for LOD hair representations. This approach uses the shading model suggested by [8] and opacity shadow maps introduced by [10]. The use of motion blurring and image blending helped to alleviate the visual artifacts associated with LOD transitions.

7.2 Performance Comparisons

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

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

for MPEGs of these simulation runs and for snapshots taken from hair simulations 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 discrete and continuous LOD representations (indicated as LODs) against (i) the use of the finest hair representations, or the lowest possible level in the hair hierarchies (indicated as Fine Strands in the table), and (ii) the coarsest strand representations, or highest level within the strand hierarchies (indicated as Coarse Strands in the table). This simulation entails wind blowing through the hair as the camera remains stationary. The camera is positioned close to the figure, so the viewer can see fine detail, and primarily shows the effects of the continuous LODs used within the strand hierarchy. Our method allows us to simulate strands with visual detail comparable to that of the finest strand representation, whereas the timings for the simulation is comparable to that of the coarsest strand representation.

Table 2 shows results of the same simulation as the camera zooms away from the figure. With this simulation the influence of the discrete LODs is obvious. The use of clusters and strips with adaptive grouping and subdivision increases the performance of the simulation with little visual loss. The table shows comparisons of the same simulation with the finest detailed strands, versus the coarsest detailed clusters and coarsest detailed strips.

For this benchmark, we used 9,350 individual strands. At the finest detail in the hierarchy, these strands were simulated with 3,570 skeletons, averaging 2.6 strands per skeleton. The algorithm does not allow all of the hierarchies to extend to each individual strand due to the overwhelming computational cost entailed. Rather, some hierarchies extend to the individual strand level, while others contain a minimum of four or five strands at the lowest level. This combination, automatically generated by our approach, enables the simulation to distribute the computational resources where they are needed the most. Hair strands that originate at the top of the head, near the part of the hair, are more viewable and will be allowed to extend to the individual strand level, whereas hairs located at the base of the neck are typically not as viewable and do not require a hierarchy reaching as far. These 9,350 strands are then represented with 110 strips, at the coarsest level, or 330 clusters at the coarsest level in the cluster hierarchy. The skeletons comprising the hairstyle consist of 6 control points on average. 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.

Breakdown	Fine Strands	LODs	Coarse Strands
Dyn Sim	0.107636	0.041624	0.038271
Col Detect	7.642328	0.411793	0.338298
Total	7.749964	0.453417	0.376569

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

Breakdown	LODs	Strands	Clusters	Strips
Dyn Sim	0.026142	0.107636	0.015374	0.003242
Col Detect	0.239489	7.642328	0.171781	0.020142
Total	0.265631	7.749964	0.187155	0.023384

Table 2. Performance Comparison. *Simulation for a camera zooming out. The average performance numbers are measured in seconds per frame.*

7.3 Analysis and Discussion

The impetus of this research is to explore the use of dynamic grouping and subdivision of hair to automatically generate continuous LODs for hair simulation. This approach enables us to further increase the visual fidelity while maintaining an interactive dynamic simulation. It is difficult to meaningfully quantify the computational errors introduced by the use of simplified representations for modeling hair. Notwithstanding the foregoing, we can subjectively evaluate the resulting simulation by performing comparisons on the visual quality of the simulated results. Using side-by-side comparisons as shown in the supplementary video, we notice higher visual fidelity of the simulated hair using continuous LODs. 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 discrete LOD representations [24].

Limitations: As with most LOD algorithms that generate hierarchical representations offline, our approach necessitates considerable memory requirements.

There are other application-dependent transition criteria, such as collision, that we have not examined closely but which can improve the system's overall performance.

7.4 Comparisons Against Earlier Approaches

This research is built upon the discrete LOD representations introduced in our recent work [24]. In [24], we discussed the benefit of using discrete LOD representations that compared favorably against earlier approaches [4, 5, 11, 12, 13, 14, 16, 17, 20, 23, 25], as the use of LODs can achieve both high visual quality and interactive dynamic simulation at the same time.

Compared to [24], our current approach allows for higher quality visual appearances while maintaining similar or better runtime performances. Using continuous and discrete LODs for hair simulation enable the user to maintain complete control over the visual and performance results of the system.

8 Summary and Future Work

In this paper, we present an approach to adaptively split and group collections of hair to generate continuous LODs for accelerating the dynamics computation of hair while achieving higher visual fidelity. In addition to potential areas of improvements mentioned in the earlier sections, there are several possible directions to extend this research:

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

Acknowledgements

This research is supported in part by Army Research Office, Intel Corporation, National Science Foundation, and Office of Naval Research.

References

[1] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. *Computer Graphics*, 26(2):111–120, 1992.
[2] D. Baraff and A. Witkin. Large steps in cloth simulation. *Proc. of ACM SIGGRAPH*, pages 43–54, 1998.
[3] F. Bertails, T.-Y. Kim, M.-P. Cani, and U. Neumann. Adaptive Wisp Tree - a multiresolution control structure for simulating dynamic clustering in hair motion. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2003.

[4] J. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. *Proc. of ACM Symposium on Computer Animation*, 2002.
[5] L. H. Chen, S. Saeyor, H. Dohi, and M. Ishizuka. A system of 3d hair style synthesis based on the wisp model. *Visual Computer*, 15(4):159–170, 1999.
[6] A. Daldegan, T. Kurihara, N. Magnenat-Thalmann, and D. Thalmann. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum (Proc. of Eurographics*, 12(3):211–221, 1993.
[7] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum (Proc. of Eurographics 2001)*, 20(3), 2001.
[8] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In J. Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 271–280, July 1989.
[9] T.-Y. Kim and U. Neumann. A thin shell volume for modeling human hair. *Computer Animation*, 2000.
[10] T.-Y. Kim and U. Neumann. Opacity shadow maps. *Proc. of Eurographics Rendering Workshop*, 2001.
[11] T.-Y. Kim and U. Neumann. Interactive multiresolution hair modeling and editing. *Proc. of SIGGRAPH*, 2002.
[12] C. K. Koh and Z. Huang. Real-time animation of human hair modeled in strip. *Eurographics CAS Workshop*, pages 101–112, 2000.
[13] C. K. Koh and Z. Huang. A simple physics model to animate human hair modeled in 2d strips in real time. *Proc. of Eurographics Workshop on Animation and Simulation*, 2001.
[14] T. Kurihara, K. Anjyo, and D. Thalmann. Hair animation with collision detection. In *Models and Techniques in Computer Animation*, pages 128–38. Springer-Verlag, 1993.
[15] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Distance queries with rectangular swept sphere volumes. *Proc. of IEEE Int. Conference on Robotics and Automation*, 2000.
[16] J. Lengyel. Real-time fur. *Proc. of Eurogrpahics Workshop on Rendering*, 2000.
[17] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. *Proc. of ACM Symp. on Interactive 3D Graphics*, 2001.
[18] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan-Kaufmann, 2002.
[19] N. Magnenat-Thalmann, S. Hadap, and P. Kalra. State of the art in hair simulation. *Int. Workshop on Human Modeling and Animaton*, pages 3–9, 2000.
[20] NVIDIA. Final fantasy technology demo 2001. *http://www.nvidia.com*, 2001.
[21] D. O'Brien, S. Fisher, and M. Lin. Simulation level of detail for automatic simplification of particle system dynamics. *Proc. of Computer Animation*, pages 210–219, 2001.
[22] E. Plante, M. Cani, and P. Poulin. Capturing the complexity of hair motion. *GMOD number 1 volume 64*, January 2002.
[23] E. Plante, M. Cani, and P. Poulin. A layered wisp model for simulating interactions inside long hair. *Proc. of Eurographics Workshop on Animation and Simulation*, 2001.
[24] K. Ward, M. Lin, J. Lee, S. Fisher, and D. Macri. Modeling hair using level-of-detail representations. *Proc. of Computer Animation and Social Agents*, 2003.
[25] Z. Xu and X. D. Yang. V-hairstudio: An interactive tool for hair design. *IEEE Computer Graphics and Applications*, 21(3):36 –43, 2001.
[26] Y. Yu. Modeling realistic virtual hairstyles. *Pacific Graphics*, 2001.

Appendix A: Implicit Integration

We use implicit integration for the dynamic simulation of hair, as explained in Section 5.6. Here we show the derivation of equations for our implicit integration formulation. We will first show how this works with the θ-component. Since we are working with polar coordinates, we will denote the angular position θ, angular velocity ω, and angular acceleration α.

We start with the second-order differential equation:

$$\ddot{\theta}(t) = f(\theta(t), \ddot{\theta}(t)) = -k_\theta(\theta_i - \theta_{i0}).$$

We can rewrite this as a first-order differential equation by substituting the variables $\alpha = \ddot{\theta}$ and $\omega = \dot{\theta}$. The resulting set of first-order differential equations is

$$\frac{d}{dt}\begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} \theta \\ \omega \end{pmatrix} = \begin{pmatrix} \omega \\ f(\theta,\omega) \end{pmatrix}$$

We get the following formulations for $\triangle\theta$ and $\triangle\omega$ when using the explicit forward Euler method, where $\triangle\theta = \theta(t_0 + h)$ - $\theta(t_0)$ and $\triangle\omega = \omega(t_0 + h)$ - $\omega(t_0)$ and h is the time step value

$$\begin{pmatrix} \triangle\theta \\ \triangle\omega \end{pmatrix} = h\begin{pmatrix} \omega_0 \\ -k_\theta(\theta - \theta_0) \end{pmatrix}$$

Instead, we are going to take an implicit step which is often thought of as taking a backwards Euler step since we are evaluating $f(\theta, \omega)$ at the point we are aiming for rather than at the point we were just at. In this case, the set of differential equations changes to the form

$$\begin{pmatrix} \triangle\theta \\ \triangle\omega \end{pmatrix} = h\begin{pmatrix} \omega_0 + \triangle\omega \\ f(\theta_0 + \triangle\theta, \omega_0 + \triangle\omega) \end{pmatrix}$$

A Taylor series expansion is applied to f to obtain the first-order approximation,

$$f(\theta_0 + \triangle\theta, \omega_0 + \triangle\omega) \approx f_0 + \frac{\partial f}{\partial\theta}\triangle\theta + \frac{\partial f}{\partial\omega}\triangle\omega$$

$$\approx -k_\theta(\theta - \theta_0) - k_\theta\triangle\theta + 0(\triangle\omega) \approx -k_\theta(\theta - \theta_0) - k_\theta\triangle\theta$$

After we substitute the approximation of f back into the differential equation we get

$$\begin{pmatrix} \triangle\theta \\ \triangle\omega \end{pmatrix} = h\begin{pmatrix} \omega_0 + \triangle\omega \\ -k_\theta(\theta - \theta_0) - k_\theta\triangle\theta \end{pmatrix}$$

We can focus on the angular velocity $\triangle\omega$ alone and substitute $\triangle\theta = h(\omega_0 + \triangle\omega)$. We get

$$\triangle\omega = h(-k_\theta(\theta - \theta_0) - k_\theta h(\omega_0 + \triangle\omega))$$

Rearranging this equation gives us

$$(1 + k_\theta h^2)\triangle\omega = -hk_\theta(\theta - \theta_0) - k_\theta h^2\omega_0$$

$$\triangle\omega = \frac{-hk_\theta(\theta - \theta_0) - h^2 k_\theta\omega_0}{1 + h^2 k_\theta}$$

Once we have calculated the change in angular acceleration, $\triangle\omega$, we can calculate the change in angular position $\triangle\theta$ trivially from $\triangle\theta = h(\omega_0 + \triangle\omega)$. The same process can be applied to the ϕ-component of the angular position and angular velocity for each control point of a strand.