Artistic Simulation of Curly Hair

Hayley Iben Mark Meyer Lena Petrovic Olivier Soares John Anderson Andrew Witkin
Pixar Animation Studios
Pixar Technical Memo #12-03a

Figure 1: *Example of stylized curly hair simulated with our method. © Disney/Pixar.*

Abstract

We present a novel method for stably simulating stylized curly hair that addresses artistic needs and performance demands, both found in the production of feature films. To satisfy the artistic requirement of maintaining the curl's helical shape during motion, we propose a hair model based upon an extensible elastic rod. We introduce a novel method for stably computing a frame along the hair curve, essential for stable simulation of curly hair. Our hair model introduces a novel spring for controlling the bending of the curl and another for maintaining the helical shape during extension. We also address performance concerns often associated with handling hair-hair contact interactions by efficiently parallelizing the simulation. To do so, we present a novel algorithm for pruning both hair-hair contact pairs and hair particles. Our method is in use on a full length feature film and has proven to be robust and stable over a wide range of animated motion and on a variety of hair styles, from straight to wavy to curly.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.3 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: Hair simulation, Mass-spring models

1 Introduction

Robustly simulating stylized hair in a production environment is a challenging problem. Hair styles can range from straight to curly and often consist of an enormous number of hair-hair interactions, leading to performance and memory concerns. In an environment driven by artistic expression, such as feature films, the shape and motion of the hairs resulting from the simulator are critical, as is the overall simulation time.

We present a hair model designed for creating specific visual looks of curly hair that are often non-physical. For example, our artists want to preserve the helical shape of the hair, regardless of intense forces caused by extreme motion only possible in animation. With a physical model for hair, such as infinitesimally thin elastic rods [Bergou et al. 2008], the helical shape would naturally straighten under such motion unless the material properties are stiffened to maintain the shape, making the hair wire-like. This natural straightening is an example of a physical motion undesired by our artists. In addition to behavioral requirements, our hair model must be robust and stable, able to handle fairly arbitrary hair shapes created by artists as depicted in Figure 1. Given the large range of motion present in an animated feature film, our simulator must be able to give dependable results without per-shot parameter tuning while preserving the overall look of the hair.

Given these requirements, we chose to represent hair as a mass-spring system defined by a piecewise linear curve that captures the deformation of the hair over time. Using a point representation for hair has several advantages. It gives artists an intuitive interface to define and modify finely detailed, complex geometry, as well as provides straightforward simulation targeting and art direction and easy integration of external forces. However, mass-spring systems have inherent limitations, most notably concerns with stability when faced with stiff systems. We ensure stability by using a semi-implicit Euler integration scheme discussed in Section 5.

In addition to the linear springs connecting particles, we introduce a spring controlling the bending along the curl and another controlling the longitudinal stretch of the curls, designed to provide our artists with the desired visual look. Because we are primarily concerned with simulating stylized curly hair, we propose a novel algorithm for computing a stable frame along the hair curve to retain the helical shape during motion. Although our method is specifically designed for simulating curly hair, we are able to handle a wide range of hair styles, as illustrated in Section 6.

Instead of simulating a dense number of hairs, our hair model uses a large number of guide hairs, each representing multiple hairs that are rendered. However, even with the reduction of hair complexity, simulating every hair-hair contact is still expensive on models with a large number of complex hairs. Instead, we present a novel algorithm for pruning the number of contacts considered by each hair, improving performance and reducing memory usage. By using this pruning, we are able to improve performance through parallelism. Without parallelization, we would be unable to meet production needs for efficiently simulating a wide range of characters.

1.1 Contributions

We present a mass-spring system with novel additions to the hair model and hair-hair contact handling so that we can robustly simulate a variety of hair styles and motion. We specifically incorporate novel components for simulating stylized curly hair into our force model. The main contributions of our paper are:

- A novel bending spring formulation using reference vectors posed in frames computed on a smoothed hair representation.
- A nonlinear core spring formulation for maintaining curly hair shape during fast motion.
- An algorithm for pruning both hair-hair contact pairs and hair particles to efficiently parallelize hair-hair contact computation.

2 Related Work

Many researchers have developed methods for modeling, dynamics, and rendering of hair in computer graphics, too numerous to adequately describe here. Instead, we describe the most relevant work to our method and refer to the survey by Ward et al. [2007] and the class notes of Bertails et al. [2008] for a broad overview.

Many methods have modeled single elastic rods based on Cosserat theory [Pai 2002; Grégoire and Schömer 2006]. Bertails et al. [2006] extended the Kirchhoff model to hair, modeling curls with a piecewise helical structure. This model contains an implicit centerline, but subsequent methods were developed with explicit centerlines for Cosserat [Spillmann and Teschner 2007] and Kirchhoff [Bergou et al. 2008; Bergou et al. 2010] models. These rod methods define material coordinate frames along the hair curve. Because we have an explicit hair representation without a predefined frame at each segment, our method parallel transports the natural Bishop frame directly, similar to the reference frame used in [Bergou et al. 2008]. However, we compute this frame along a smoothed representation of the hair curve instead of the curve itself, reducing the sensitivity of the frame to changes in hair positions.

Much previous work has applied mass-spring systems to individual hairs. One of the first approaches was Rosenblum et al. [1991], which used a linear spring for stretch and an angular spring between segments for bend. Petrovic et at. [2005] used a mass-spring system for simulating the dynamics of keyhairs that represent multiple rendered hairs. Selle et al. [2008] presented a mass-spring model for simulating all individual hairs. They used separate edge, bend, twist, and altitude springs to form an implied tetrahedron of springs between points, preventing volume collapse. Similar to these methods, we use a linear spring for stretch. Our hair model differs with the usage of two additional springs, designed to give our artists the visual look described in Section 1. We add a single spring for controlling bend, using the stably generated frame discussed above for the hair orientation. We define an additional spring to control the longitudinal stretch of curls during motion, not present in prior models.

A variety of methods have been proposed to model hair volume and hair-hair contacts. Hadap et al. [2001] modeled hair as a serial rigid multibody chain and incorporated fluid dynamics to model hair collisions and contacts. Plante et al. [2002] constrained a mass-spring system to a deformable envelope defining a volume of the cluster of hairs (wisp), which was also used for interactions. Bando et al. [2003] model the hair as a set of particles with density representing the sampled hair volume. Choe et al. [2005] combined a mass-spring model with a serial rigid multibody chain to model wisps, detecting contacts through cylinders. Hadap [2006] further extends the rigid multibody model to include tree structures by solving with differential algebraic equations, more easily allowing analytic constraints.

Mass-spring models have also been combined with a lattice for deformation during hair styling [Gupta et al. 2006] or with an Eulerian fluid solver to give volume and provide a better initial position for the particle contacts [McAdams et al. 2009]. More recently, hair-body and hair-hair contacts have been more accurately modeled by using a non-smooth Newton solver for the Coulomb friction law [Bertails-Descoubes et al. 2011]. Daviet et al. [2011] globally solves for Coulomb friction with a hybrid Gauss-Seidel algorithm, using an analytic solver to ensure convergence.

Similar to prior work, we chose to detect contacts by surrounding particles with geometry to preserve volume, in our case spheres, and use penalty forces to handle interactions. Our method differs from previous work by introducing a novel algorithm to prune both hair-hair pairs and hair particles to improve performance by allowing parallelization while still producing good results. We chose the inaccurate yet efficient penalty force model to locally solve contacts for performance reasons instead of handling contacts globally, such as in continuum methods or solving exact Coulomb friction. Although penalty forces can cause instabilities, using a semi-implicit integration method combined with tapering spring constants during contact release produces stable results.

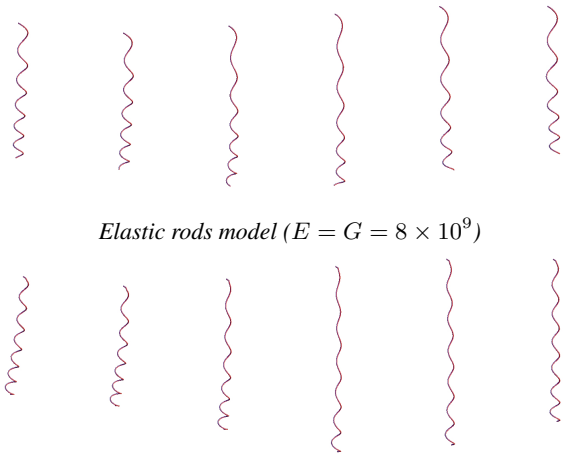
3 Hair Model

Our approach models hair as a infinitesimally thin elastic rod, similar to prior work (eg. [Bergou et al. 2008; Bergou et al. 2010]). We represent a single hair by a piecewise linear curve and calculate a material frame for each point. Our force model must satisfy three properties to create our desired visual looks. First, we desire relatively inextensible hair rods, allowing artistic control over stretch. Second, we want a hair model that holds an arbitrary curl's shape while allowing it to flex with motion. Last, we want our curls to reasonably maintain their initial shape during motion, avoiding curl unwinding.

We model these properties for a single hair by using a mass-spring system. Similar to [Rosenblum et al. 1991; Petrovic et al. 2005; Selle et al. 2008; McAdams et al. 2009], we sequentially connect each hair particle with a linear spring, controlling stretch (Section 3.1). Our method differs from prior work with the formulations of two additional springs. To control bend along the curl, we propose a novel bending spring by computing frames along a smoothed representation of the curve (Section 3.2). We also introduce a spring limiting the longitudinal stretch of curls (Section 3.3). These three springs comprise our per hair force model.

3.1 Stretch Spring

We define our hair model using a set of particle positions connected by linear springs. Let each hair be defined by a set of current particle positions $P = \{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}\}$ and initial rest pose particles, $\bar{P}$, where $\bar{\cdot}$ denotes rest quantities. Let the current velocities for



Elastic rods model ($E = G = 8 \times 10^9$)

Our bending model ($k_b = 3 \times 10^4$, $c_b = 1.74 \times 10^3$)

Figure 2: *Simple example of a perfect helical curl undergoing motion from the walk cycle of Figure 14. The elastic rod model (top) introduces rotation around the helix, apparent in the orientation of the end of the curl, while our bending model (bottom) does not.*

these particles be $V = \{\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{N-1}\}$ and the polyline edges connecting hair particles be $\boldsymbol{e}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$. We compute a standard linear spring force on particle i by

$$\boldsymbol{f}_s(k_s, c_s)_i = k_s(\| \boldsymbol{e}_i \| - \| \bar{\boldsymbol{e}}_i \|)\hat{\boldsymbol{e}}_i + c_s(\Delta \boldsymbol{v}_i \cdot \hat{\boldsymbol{e}}_i)\hat{\boldsymbol{e}}_i \qquad (1)$$

where k_s are the spring and c_s the damping coefficients, $\Delta \boldsymbol{v}_i = \boldsymbol{v}_{i+1} - \boldsymbol{v}_i$, $\| \cdot \|$ denotes vector length and $\hat{\cdot}$ vector normalization. Because springs in our model are connecting two particles, each of our spring forces is applied equally to both particles in opposite directions.

Our artists also want to enable bounce during a walk cycle, for example, by allowing hair to slightly stretch (see Figure 14 and the video for an example). To allow this artistic stretch of the hair without using stiff springs, we impose an upper limit on the stretch of the polyline similar to the biased strain limiting approach presented in [Selle et al. 2008]. During the spring damping calculation, we recurse from the root to the tip limiting the velocity of the hair particle if $\Delta \boldsymbol{v}_i^2$ exceeds a threshold. After we update positions, we again recurse from the root of the hair to the tip shortening edges that exceed the specified stretch allowance. By using this limiter, artists are able to control the desired amount of stretch allowed by the system.

3.2 Bending Spring

Our method is most similar to the elastic rods model [Bergou et al. 2008; Bergou et al. 2010], which computes the material frame as the minimizer of elastic energy of the curve. They use these material frames to compute the bending and twisting energies along the rod. However, this formulation introduces rotation during simple motion of a curl, such as a walk cycle, shown in Figure 2 *top*. This rotation can be removed by increasing the material's stiffnesses, but this change leads to wire-like behavior (see video).

This type of rotation in the curl, while physically accurate, is undesired for our hair model. We instead introduce a novel method to stably generate the material frame by parallel transporting the root frame of the hair along a smoothed piecewise linear curve (see Section 3.2.1). The result of using our smoothed curve is a series

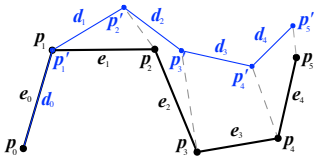


Figure 3: *Original curve (bold line) with points $\boldsymbol{p}_i$ and edges $\boldsymbol{e}_i$ overlaid with corresponding smoothed curve (thin blue line) with points $\boldsymbol{p}'_i$ and edges defined by vectors $\boldsymbol{d}_i$.*

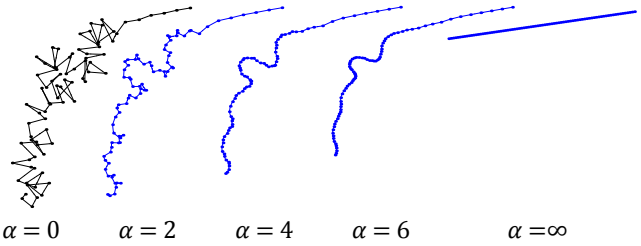


Figure 4: *Example of a stylized curly hair (far left) and the smoothed curves (blue) computed with α at 2, 4, 6, and ∞.*

of frames that are not highly influenced by small changes in point positions, avoiding unwanted rotation as illustrate by Figure 2 *bottom*. Our bending formulation uses reference vectors posed in the frames from the smoothed curve to compute the bending force (see Section 3.2.2).

3.2.1 Smoothing Function

Let $\Lambda = \{\boldsymbol{\lambda}_0, \ldots, \boldsymbol{\lambda}_{N-1}\}$ be a set of N elements in $\mathbb{R}^3$ associated with a hair, such as particle positions or velocities. We define our smoothing function, $\boldsymbol{d}_i = \boldsymbol{\varsigma}(\Lambda, \alpha)_i$, with an infinite impulse response (IIR) Bessel filter (for examples, see [Najim 2006]). These filters are recursive functions, combining the input and prior results to produce each new result.

To compute the results, we take as input the smoothing amount, $\alpha \geq 0$, in units of length along the curve. Let $\beta = \min(1, 1 - \exp(-l/\alpha))$ where l is the average rest length per segment of the hair being smoothed. We then recursively compute vectors $\boldsymbol{d}_i$ $\forall i \in [0 \ldots N-2]$ with the equation

$$\boldsymbol{d}_i = 2(1-\beta)\boldsymbol{d}_{i-1} - (1-\beta)^2\boldsymbol{d}_{i-2} + \beta^2(\boldsymbol{\lambda}_{i+1} - \boldsymbol{\lambda}_i) \quad . \qquad (2)$$

By choosing these coefficients and initializing $\boldsymbol{d}_{-2} = \boldsymbol{d}_{-1} = \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0$, this equation reduces to $\boldsymbol{d}_0 = \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0$ at $i = 0$. Subsequent $\boldsymbol{d}_i$ values are weighted towards this initial direction, an important property for our usage of this function.

When Λ is a set of positions, we can reconstruct the smoothed polyline by recursively adding the vectors from the fixed root position. The new points, $\boldsymbol{p}'_i$ $\forall i \in [1 \ldots N-1]$, are defined by

$$\boldsymbol{p}'_i = \boldsymbol{p}'_{i-1} + \boldsymbol{d}_{i-1} \qquad (3)$$

where $\boldsymbol{p}'_0 = \boldsymbol{\lambda}_0$, the root of the polyline. We show the relationship between an original and smoothed curve in Figure 3.

The range of input values $\alpha = [0, \infty]$ produces well behaved output from Equation 2. If $\alpha = 0$, then we set $\beta = 1$, meaning that $\boldsymbol{d}_i = \boldsymbol{\lambda}_{i+1} - \boldsymbol{\lambda}_i$ and no smoothing occurs. As $\alpha \to \infty$, then $\beta \to 0$ and $\boldsymbol{d}_0 = \cdots = \boldsymbol{d}_{N-2} = \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0$. If this limit case occurs when Λ is a set of positions, the polyline resulting from Equation 3 is straight and in the direction of the first segment, regardless of

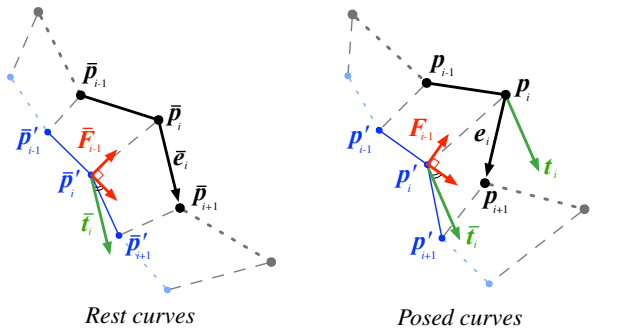


Figure 5: *We precompute the rest edge $\bar{\boldsymbol{e}}_i$ (left) and store it in the local frame $\bar{\boldsymbol{F}}_{i-1}$ (show at $\bar{\boldsymbol{p}}_i$ for illustration) giving the reference vector $\bar{\boldsymbol{t}}_i$. We use the current pose's local frame $\boldsymbol{F}_{i-1}$ (right) to compute $\boldsymbol{t}_i$, our target bending direction, from $\bar{\boldsymbol{t}}_i$. Note that bold black line represents the hair while the thin blue line represents the smoothed curve.*

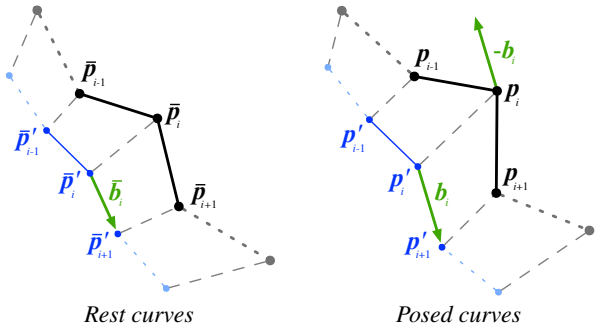


Figure 6: *We precompute the rest core directions $\bar{\boldsymbol{b}}_i$ (left) that we use to compute the spring force and apply in the current pose, in the direction $-\boldsymbol{b}_i$ (right).*

how kinked the input polyline. Figure 4 gives an example curve from the model depicted in Figure 1 and the resulting smooth curve with different α values.

3.2.2 Bending Spring Formulation

We implement a bending spring force to stably control the bend between the rest and current poses of the hair while maintaining the helical shape of curls. During initialization, we use the rest pose points to precompute a reference vector, $\bar{\boldsymbol{t}}_i = \bar{\boldsymbol{F}}_{i-1}^T \bar{\boldsymbol{e}}_i$, as the edge $\bar{\boldsymbol{e}}_i$ expressed in the local frame, $\bar{\boldsymbol{F}}_{i-1}$, of point $\bar{\boldsymbol{p}}_{i-1}$. We store the axes of the frame as the columns of $\bar{\boldsymbol{F}}_{i-1}$ and use $\bar{\boldsymbol{e}}_i$ as a column vector. We compute the local frames, $\bar{\boldsymbol{F}}_{i-1}$, by parallel transport of the root rest frame, $\bar{\boldsymbol{F}}_0$, along the smoothed curve defined by $\varsigma(\bar{P}, \alpha_b)$ with α_b bending smoothing amount (see Figure 5 *left*).

We use the common approach of rooting hair to a planar scalp polygon, which provides the animated motion for the hair and the hair's root coordinate frame, $\boldsymbol{F}_0$. At each step of the simulation, we compute $\boldsymbol{F}_0$ from the current root polygon and $\boldsymbol{F}_{i-1}$ from $\varsigma(P, \alpha_b)$. We use these local frames to stably pose the stored reference vectors $\bar{\boldsymbol{t}}_i$ in the current hair configuration. The resulting vectors are the target vectors, $\boldsymbol{t}_i = \boldsymbol{F}_{i-1}\bar{\boldsymbol{t}}_i$, that we want our current pose to match, as illustrated in Figure 5 *right*. We add a spring force between the edge in the current pose, $\boldsymbol{e}_i$, and the target vector, $\boldsymbol{t}_i$, to obtain the bending force

$$\boldsymbol{f}_b(k_b, c_b)_i = k_b(\boldsymbol{e}_i - \boldsymbol{t}_i) + c_b(\Delta\boldsymbol{v}_i - (\Delta\boldsymbol{v}_i \cdot \hat{\boldsymbol{e}}_i)\hat{\boldsymbol{e}}_i) \quad (4)$$

where k_b is the spring and c_b the damping coefficients. To provide more flexibility, we allow artists to specify the spring coefficients for each edge. This bending formulation preserves linear momentum but not angular momentum. We could extend the model to also preserve angular momentum but have found it unnecessary for our application.

3.3 Core Spring

Using only stretch and bending springs were insufficient for our artists' needs when simulating curly hair over a variety of motions. A mass-spring model with stiff stretch and bending springs can hold the curly shape during fast motion, but it reduces the ability for the hair to bend. If we instead reduce the stiffness of the bending springs to allow flexibility, the curls lose shape and unwind at high accelerations. The resulting curl becomes nearly straight and appears much longer than the desired look, even though there is minimal stretch. Such behavior is natural for physical models, such as elastic rods [Bergou et al. 2008; Bergou et al. 2010]. However, this curl unwinding is undesired behavior for our artistic needs. To allow flexible curly hair yet maintain shape, we introduce a third *core* spring that controls the longitudinal stretch of the curls without stiffening bending springs. See Figure 10 and the accompanying video for examples.

Similar to the bending calculation, we create a smoothed representation of the hair using the function described in Section 3.2.1. We precompute the representation of the original core of the curl from the rest points, $\bar{\boldsymbol{b}}_i = \varsigma(\bar{P}, \alpha_c)_i$, where α_c is the smoothing amount for core springs (see Figure 6 *left*). The resulting piecewise linear curve represents the direction of the center of the curl, giving a more natural representation to control its longitudinal stretch, as illustrated by Figure 3.

During the force calculation, we use the current hair pose to compute $\boldsymbol{b}_i = \varsigma(P, \alpha_c)_i$ (see Figure 6 *right*). We apply the same smoothing to the velocities to obtain the vectors $\boldsymbol{\nu}_i = \varsigma(V, \alpha_c)_i$ for the spring damping term. We compute the core spring force by calculating the amount of stretch along the core, giving the force

$$\boldsymbol{f}_c(k_c, c_c)_i = k_c(\| \boldsymbol{b}_i \| - \| \bar{\boldsymbol{b}}_i \|)\hat{\boldsymbol{b}}_i + c_c(\boldsymbol{\nu}_i \cdot \hat{\boldsymbol{b}}_i)\hat{\boldsymbol{b}}_i \quad (5)$$

where k_c is the spring and c_c the damping coefficients. To maintain stability, we must keep $k_c < k_s$ where k_s is the spring coefficient maintaining the connection between hair particles from Equation 1.

Because this spring is intended to control longitudinal stretching, we avoid adding unnecessary constraints by allowing the core to compress. We set the spring coefficient k_c to zero during compression, determined from the spring length ($\| \boldsymbol{b}_i \| - \| \bar{\boldsymbol{b}}_i \|$). We blend from zero to its full value upon extension to avoid a large force introduction. We empirically found that using a cubic Hermite blend function from $[0, 0.5]$ is sufficient because the segments of our hair lengths are on the order of 1 unit of length.

During extreme motion (see Figure 10 and the video), a high core stiffness value is required to keep curls from unwinding. However, tuning k_c for this extreme motion would undesirably stiffen the curl's shape during normal motion, such as a character's walk cycle. To balance the artist's desire for bounce during the walk cycle and control during extreme motion, we nonlinearly change k_c from the current value to a maximum specified stiffness using a cubic Hermite blend function based upon the amount of longitudinal curl stretch. We compute the blend weight from the ratio of the total length of the current curve, $\boldsymbol{b}$, to the total length of the rest curve, $\bar{\boldsymbol{b}}$. By automatically adjusting the core stiffness and damping, we avoid the need for per-shot parameter tuning.

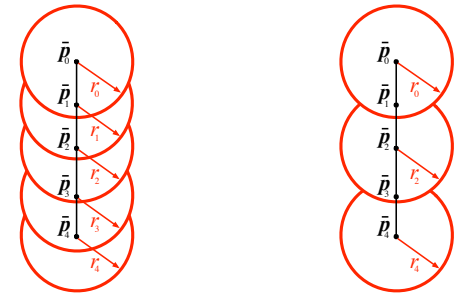


Figure 7: *Each particle $\bar{\mathbf{p}}_i$ has an associated sphere with radius r_i (left). We prune particles contained in overlapping spheres (right) to reduce the number of potential contacts.*

4 Hair-Hair Contacts

In addition to modeling a single hair, we must also model hair-hair contacts so that we can capture the interesting interactions when simulating a mass of hair. If we were to consider every possible pair of particles for interaction, the algorithm would become quite expensive. Instead, we reduce the number of interactions considered by performing two kinds of pruning, enabling us to parallelize our hair simulation and improve performance.

We prune contact points along the hair, as described in Section 4.1. We also prune hair pairs allowed to interact during the simulation. At initialization, we build a hair contact graph C containing an edge $C(h_i, h_j)$ between every pair of hairs. Our hair pair pruning then removes edges from the graph, as described in Section 4.2. This pruning enables us to parallelize our simulation and optimize the communication pattern between processors.

We could have alternatively used a bounding hierarchy to accurately handle contacts in all situations. The advantage of our method over such hierarchies is that we can statically compute the processor communication pattern; otherwise, we would need to dynamically update the processor communication or exchange all hair data between processors. The disadvantage of our algorithm is that the contacts may be incorrect in certain situations, such as when long hair is flipped from one side of the scalp to the other. For our production needs, we have run into few examples where this limitation has caused artifacts. In these rare cases, we decrease pruning to guarantee the correct contacts.

4.1 Pruning Hair Particles

Our hair-hair contact model must account for volume between guide hairs, each representing several rendered hairs. We use spheres around individual hair particles to indicate the volume each particle represents. Increasing these sphere radii increases the volume of the hair. Because spheres of neighboring particles on the same hair overlap, we can reduce particles used for contact testing by pruning those contained inside neighboring spheres (see Figure 7 *left*). Even with hair motion, the spheres around neighboring particles will provide enough contact for hair-hair interactions.

Our pruning test starts at the root of the hair ($j = 0$), which we never prune, and compares the sum of the edges $\bar{\mathbf{e}}_i$ to the potentially overlapping spheres. We set $k = 1$ and continue to increment k until the following equation is satisfied:

$$\sum_{i=j}^{k-1} \| \bar{\mathbf{e}}_i \| > s(r_j + r_k) \tag{6}$$

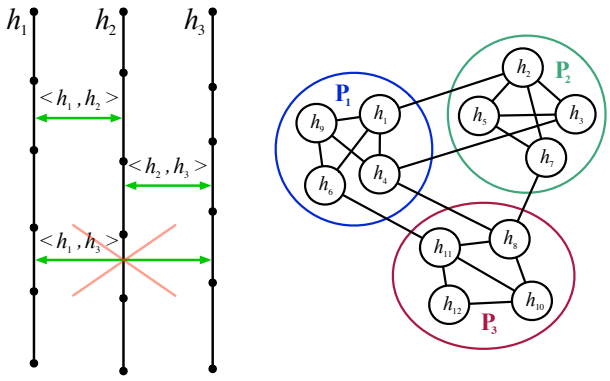


Figure 8: *By pruning hair pairs (simple example on left), we reduce the overall number of edges in the larger contact graph C (right). This reduction enables us to cluster the graph into processors with reduced communication cost between P_1, P_2, and P_3.*

where r_j is the radius for the j^{th} particle and s a parameter controlling particle sparsity. We then prune all particles between j and k, set $j = k$, $k = k + 1$ and repeat the process until we have reached the end of the hair. This process controls how much overlap there is between the contact spheres after pruning. Setting $s = 1$ means that spheres would just touch with no overlapping. We have found that a value of 0.75 provides a good balance between performance and accuracy. The result is a subset of available particles for handling contacts, as illustrated in Figure 7 *right*.

4.2 Pruning Hair Pairs

Hair-hair contact models are used to represent forces such as static charge and friction among neighboring hairs. We have observed that when many hairs interact in a complex manner, it is not necessary to capture all of the individual hair-hair interactions in order to produce a plausible, visually appealing result. In fact, we can ignore certain hair-hair interactions and their effect will often be felt through similar or indirect interactions with neighboring hairs. For example, consider three hairs h_1, h_2 and h_3 that are aligned in a row, as in Figure 8 *left*. We can prune the interaction $< h_1, h_3 >$ and assume that the effect of this interaction will happen through contacts between $< h_1, h_2 >$ and $< h_2, h_3 >$. As the numbers of hairs increase, the effects of these individual hair-hair interaction become less important as we mainly see the aggregate effects of many hair-hair interactions. Using this observation, we have developed an algorithm to prune hair pairs used for contact testing - effectively sampling the hair-hair interactions. This pruning allows us to reduce the number of necessary hair-hair contacts, reduce interprocessor communication and more efficiently parallelize the simulation.

To indicate the hairs allowed to interact during simulation, we use a graph C where each node is a hair and an edge $C(h_i, h_j)$ indicates that hairs h_i and h_j are allowed to interact. We statically prune our contact graph at initialization time as follows: First, we multiply each r_i by a constant, r_c, and compute and sum all contacts between spheres on hair i and hair j, calling the sum $n_{i,j}$. We prune edges where $n_{i,j} < n_t$, a threshold indicating the minimum number of hair contacts required for interaction. We also stochastically prune graph edges based on our observation that neighboring hairs will handle the interactions. It is important to note that we are pruning hair-hair interactions from the graph; we are not pruning the hairs themselves.

Once we have pruned our graph, we assign hairs to processors for

parallelization. To optimize performance, we attempt to minimize the data exchanged between processors by using a greedy graph clustering algorithm. We initially assign equal numbers of hairs to each processor, creating a graph clustering. The goal is to reduce the communication cost, computed as the number of edges between the processor groups, while maintaining equal workloads. To do so, we greedily swap hairs between processors if it reduces the communication cost, iterating until we have reached a minimum, or have exceeded a maximum number of swaps (see Figure 8, *right*). This final clustering allows us to send less information between processors than if we had simply used all contact pairs, leading to more efficient parallelization. Note that we could use any algorithm for constructing the hair adjacency graph as long as it produces good communication patterns between processors.

4.3 Contact Detection and Response

At each step of our simulation, we spatially subdivide our scene into a uniform grid structure where we insert the hair particles available for contact. For each hair particle, we use the grid to retrieve neighboring hair particles for contact testing. We discard potential contacts prior to testing if their is no hair pair edge $C(h_i, h_j)$ between the hairs containing the particles.

If the hair pair $< h_i, h_j >$ has not been discarded, we detect contacts when their spheres overlap, $\| \boldsymbol{p}_i - \boldsymbol{p}_j \| < r_i + r_j$. Similar to prior methods [Chang et al. 2002; Bando et al. 2003; Selle et al. 2008], we handle interactions by applying a spring penalty force and break contacts when particles surpass a distance threshold. Our method dynamically creates contacts, similar to [Selle et al. 2008]. However, we attach springs between particles instead of edges and do not directly limit the overall number of allowed contacts. Instead, we have already limited our contacts through pruning particles and hair pairs.

To avoid instabilities often associated with penalty forces, such as high-frequency bouncing of the objects in contact, we adjust spring constants when the spring breaks using a method similar to [Chang et al. 2002] but with some additions. We dynamically increase the spring stiffness during initial contact while using full damping to avoid a large spring impulse. As the spring breaks, we first decrease the stiffness to zero followed by the damping. We allow the spring constants to reengage if the particle's distance decreases before the spring breaks, so that the contacts do not pass through one another.

5 Implementation Details

We ensure stability by using a nested semi-implicit Euler integration scheme with fixed time steps. Although one could possibly use adaptive time stepping, we have found it unnecessary in practice. We detect collisions and contacts in our outer loop. Our first nested loop (the force loop) integrates the hair model springs, collision and contact forces, while a further nested loop (the damping loop) integrates the hair model spring damping forces. Our algorithm can be summarized a follows:

```
For each outer loop iteration:
    Detect Collision and Contacts

    For each force loop iteration:
        Integrate Internal Hair Forces (Section 3)
        Integrate External Forces
        Handle Hair-Hair and Hair-Object Collisions

        For each damping loop iteration:
            Integrate Damping Forces

    Update Positions
    Exchange Hair Data Between Processors
```

Figure 9: *Example of stylized curly hair.* © *Disney/Pixar.*

For all models except RedHead, the outer loop time step is 0.00138883 seconds (24 frames per second with 30 outer loop steps per frame), the first nested loop time step is 9.25887e-05 (15 force steps per outer step), and second nested loop time step is 9.25887e-06 (10 damping steps per force step). For RedHead, we cut the outer time step in half to 0.000694416 (60 outer steps per frame) to handle the higher spring stiffnesses and used the same number of force and damping steps given above (force loop: 15 steps, 4.62944e-05 seconds per step; damping loop: 10 steps, 4.62944e-06 seconds per step). After determining the time steps for the models, no additional time step tuning was needed during production shots as this integration scheme provided consistent and stable results.

In our implementation, we use MPI for inter-processor communication. For hair-object collision detection, we spatially subdivide the scene into a uniform grid using a method similar to [Teschner et al. 2003]. Our simulator uses penalty forces for collision response, a commonly used approach [Plante et al. 2002; Choe et al. 2005; Bertails et al. 2006; McAdams et al. 2009] that is fast to compute. However, our contributions do not rely on which collision model we use and an alternative method could be substituted.

6 Results and Discussion

Our simulator is in use by a feature film with a variety of characters and hair styles. By creating a stable simulator, we are able to generate most simulations using our default character parameters (examples of the default parameters for our RedHead and Horse characters are given in Table 1). These parameters are tuned once per character, starting from the appropriate default. Eliminating per-shot parameter tweaking provides a dramatic benefit to production schedules and can have a great impact on the production budget. Although many of the shots are "out of the box", our simulator provides additional controls and supports external forces for artistic direction (a common use is to keep the characters' hair from blocking their face).

Effects of Bending Smoothing: Figure 2 and the accompanying video shows the importance of using our smoothed hair for bending frame propagation on a simple helical curl example. Our method avoids unwanted rotations in the curl present in physically accurate models such as elastic rods. We actively choose the physical in-

Symbol	RedHead	Horse
k_s	$5 \times 10^6 - 1 \times 10^7$	3×10^5
c_s	$4472 - 4743$	6025
α_b	10	$0 - 1$
k_b	$100 - 7.2 \times 10^4$	$300 - 4.7 \times 10^4$
c_b	$40 - 2495$	$22 - 1083$
α_c	3	0
k_c	$1.5 \times 10^4 - 6 \times 10^5$	0
c_c	$100 - 1.0 \times 10^4$	0
r_i	$0.23 - 1.65$	$1.0 - 2.5$
r_c	5	$3 - 4$
n_t	1	$3 - 20$

Table 1: *Typical range of parameters for the models.*

accuracy of our smoothed bending model specifically to provide a smooth and consistent motion and orientation of the curl, desired for our artistic setting.

Effects of Core Springs: The accompanying video shows the importance of core springs. Even in a simple walk cycle, without core springs the hair sags unnaturally. This sag can be removed by increasing the bending springs, but this results in an unnaturally stiff looking hair style. Core springs allow us to maintain the curl shape without making the hair unnecessarily stiff.

The need for core springs is even greater during fast motions and high accelerations as shown in Figure 10 and the video. Without core springs, when the character stops abruptly, the high acceleration causes the curls to unwind. It is important to note that the length of the hair segments themselves are not extending much here, rather the curls are unwinding, resulting in a visually longer hair. Using core springs, the curls maintain the shape much better. These extreme, non-physical accelerations are not uncommon when dealing with animated films.

Parallelization: Table 2 shows the average seconds per frame for different simulations using our system. Notice that our system achieves good speedups as we increase the number of processors (from around 4.5x - 6x on 10 processors). For some examples, the performance gains reduce or even become negative around 12 processors. This is due to the communication cost beginning to exceed the gains of using more cores - usually because there are not enough hair points on each processor. Communication costs are even more dominant if we do not use our hair-hair contact pruning. When not using our pruning, performance drops by a factor of 1.4x - 4.1x depending on the example. With simulation times of 13.4 seconds per frame for our RedHead hero character and her complex hair, our simulator has proved to be extremely efficient in our production process.

Production results: Our system was initially designed to deal with the challenges of our RedHead hero character and her complex, stylized curly hair (as seen in Figures 1 and 9). Figure 14 and the video show the character during a walk cycle, while Figure 15 and the video show the character performing different head rotations from a calisthenic set. Notice how the hair motion achieves a nice bounce and has complex hair-hair interactions with a natural tumbling and flow as desired by our artists.

Although our simulator was designed to handle the challenges of RedHead's curly hair, it is more than capable of simulating other styles. Figure 11 and the video shows our system applied to the various lengths of straight hair for a horse (mane, tail, fetlocks, etc.) during a walk cycle. Figure 16 and the video shows the horse during a more agitated side-stepping motion. Figures 12 and 13 and the remainder of the video show the wide range of hairstyles simulated with our system - from curly, to wispy, to straight.

Without core springs *With core springs*

Figure 10: *Maximum extension of curls without core springs (left) and with core springs (right). Because the hair without core springs is unable to maintain the curl and continues to unwind, that pose occurs 3 frames later (at 24 fps) than the example on the right.* © *Disney/Pixar.*

Figure 11: *Example of various lengths of straight hair styles for a horse (mane, tail, fetlocks, body).* © *Disney/Pixar.*

7 Conclusions and Future Work

We have shown that our system provides efficient simulation of curly hair while being able to realize our artistic goals. This was achieved through several key contributions. Our novel smoothed bending formulation produces visually pleasing bends and twists while removing unwanted twist discontinuities present in physical rods. Our nonlinear core spring force resists unwinding and maintains curl shape, without making the hair unnecessarily stiff, even during extreme motion. The hair-hair contact pruning allows us to efficiently parallelize the simulation by reducing the interprocessor communications while still achieving plausible dynamics. Our simulator is in use on a full-length feature film on a range of characters and hair styles and has proven to be an efficient and indispensable part of our production process.

The main limitation of our work is due to our hair-hair contact pruning, we may actually miss contacts between the hairs. In all but a few extreme cases (such as when a character flips all of the hair from one side of their head to another), we have not seen any artifacts from this as the effect of this contact is usually handled by one of the contact pairs that have not been pruned. In the cases where pruning will cause a problem (such as the aforementioned hair flip), we use our pruning controls to reduce the amount of pruning, trading speed for accuracy.

Although our simulator is quite efficient, we hope to find ways to improve its performance. In future work, we plan to look at other schemes for processor assignment from the contact graph. This may be even more important as we also plan to examine the effects of implementing our system in a heterogeneous environment (across several CPUs, GPUs, etc.) as the communication costs between those devices will vary.

Model	Hairs	Points	Number of Processors						
			1	2	4	6	8	10	12
RedHead	579	44,552	98.9	50.5	28.9	20.9	17.3	16.6	13.4
Horse (total)	9,700	117,607	125.45	72.0	47.0	39.2	27.6	25.3	23.2
Mane	2,604	43,591	41.2	26.3	19.1	16.9	8.9	9.0	6.9
Tail	425	18,275	40.8	22.6	14.7	12.1	10.1	8.8	9.0
Fetlocks (Front)	996	17,928	14.6	7.9	4.4	3.4	2.8	2.4	2.3
Fetlocks (Rear)	726	13,068	9.95	5.4	3.2	2.6	2.2	1.9	2.0
Body	4,949	24,745	18.9	9.8	5.6	4.2	3.6	3.2	3.0

Table 2: *Average frame time (in seconds) for the walk cycle of our RedHead character (averaged over 232 frames) and Horse character (averaged over 119 frames). Simulations and timings were run on dual X5660 2.80GHz processor machine with 12 cores and 24 GB memory.*

Figure 14: *Example of a walk cycle of our stylized curly hair character. © Disney/Pixar.*

Figure 15: *Example of volume preservation during head rotation. © Disney/Pixar.*

Figure 16: *Example of horse hair with large side stepping motion. © Disney/Pixar.*

Figure 12: *Example of other hair styles simulated with our method. © Disney/Pixar.*

References

BANDO, Y., CHEN, B.-Y., AND NISHITA, T. 2003. Animating hair with loosely connected particles. *Computer Graphics Forum 22* (Sept.), 411–418.

BERGOU, M., WARDETZKY, M., ROBINSON, S., AUDOLY, B., AND GRINSPUN, E. 2008. Discrete elastic rods. In *ACM SIGGRAPH 2008 Papers*, 63:1–63:12.

BERGOU, M., AUDOLY, B., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2010. Discrete viscous threads. In *ACM SIGGRAPH 2010 papers*, 116:1–116:10.

BERTAILS-DESCOUBES, F., CADOUX, F., DAVIET, G., AND ACARY, V. 2011. A nonsmooth newton solver for capturing

Figure 13: *Example of another style of curly hair. © Disney/Pixar.*

exact coulomb friction in fiber assemblies. *ACM Trans. Graph. 30* (February), 6:1–6:14.

BERTAILS, F., AUDOLY, B., CANI, M.-P., QUERLEUX, B., LEROY, F., AND LÉVÊQUE, J.-L. 2006. Super-helices for predicting the dynamics of natural hair. In *ACM SIGGRAPH 2006 Papers*, 1180–1187.

BERTAILS, F., HADAP, S., CANI, M.-P., LIN, M., KIM, T.-Y., MARSCHNER, S., WARD, K., AND KAČIĆ-ALESIĆ, Z. 2008. Realistic hair simulation: animation and rendering. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, 89:1–89:154.

CHANG, J. T., JIN, J., AND YU, Y. 2002. A practical model for hair mutual interactions. In *Proc. of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 73–80.

CHOE, B., CHOI, M. G., AND KO, H.-S. 2005. Simulating complex hair with robust collision handling. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 153–160.

DAVIET, G., BERTAILS-DESCOUBES, F., AND BOISSIEUX, L. 2011. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In *Proc. of the 2011 SIGGRAPH Asia Conference*, 139:1–139:12.

GRÉGOIRE, M., AND SCHÖMER, E. 2006. Interactive simulation of one-dimensional flexible parts. In *Proc. of the 2006 ACM symposium on Solid and Physical Modeling*, 95–103.

GUPTA, R., MONTAGNOL, M., VOLINO, P., , AND MAGNENAT-THALMANN, N. 2006. Optimized framework for real time hair simulation. In *Proc. of Computer Graphics International (CGI'06), LNCS*, 702–710.

HADAP, S., AND MAGNENAT-THALMANN, N. 2001. Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 329–338.

HADAP, S. 2006. Oriented strands: dynamics of stiff multi-body system. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 91–100.

MCADAMS, A., SELLE, A., WARD, K., SIFAKIS, E., AND TERAN, J. 2009. Detail preserving continuum simulation of straight hair. In *ACM SIGGRAPH 2009 Papers*, 62:1–62:6.

NAJIM, M., Ed. 2006. *Digital Filters Design for Signal and Image Processing*. Wiley-ISTE.

PAI, D. K. 2002. Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum 21*, 3 (Sept.), 347–352.

PETROVIC, L., HENNE, M., AND ANDERSON, J. 2005. Volumetric methods for simulation and rendering of hair. Technical Memo 06-08, Pixar Animation Studios.

PLANTE, E., CANI, M.-P., AND POULIN, P. 2002. Capturing the complexity of hair motion. *Graphical Models 64*, 1 (Jan.), 40–58.

ROSENBLUM, R., CARLSON, W., AND TRIPP III, E. 1991. Simulating the structure and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation 2*, 141–148.

SELLE, A., LENTINE, M., AND FEDKIW, R. 2008. A mass spring model for hair simulation. In *ACM SIGGRAPH 2008 Papers*, 64:1–64:11.

SPILLMANN, J., AND TESCHNER, M. 2007. Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Proc. of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 63–72.

TESCHNER, M., HEIDELBERGER, B., MUELLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. In *Vision, Modeling and Visualization*, 47–54.

WARD, K., BERTAILS, F., KIM, T.-Y., MARSCHNER, S. R., CANI, M.-P., AND LIN, M. C. 2007. A survey on hair modeling: Styling, simulation, and rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (Mar.), 213–234.