A Statistical Wisp Model and Pseudophysical Approaches for Interactive Hairstyle Generation

Byoungwon Choe and Hyeong-Seok Ko
Seoul National University

Abstract— This paper presents an interactive technique that produces static hairstyles by generating individual hair strands of the desired shape and color, subject to the presence of gravity and collisions. A variety of hairstyles can be generated by adjusting the wisp parameters, while the deformation is solved efficiently, accounting for the effects of gravity and collisions. Wisps are generated employing statistical approaches. As for hair deformation, we propose a method which is based on physical simulation concepts but is simplified to efficiently solve the static shape of hair. On top of the statistical wisp model and the deformation solver, a constraint-based styler is proposed to model artificial features that oppose the natural flow of hair under gravity and hair elasticity, such as a hairpin. Our technique spans a wider range of human hairstyles than previously proposed methods, and the styles generated by this technique are fairly realistic.

Index Terms— *hair modeling, Markov chain, statistical wisp model, hair deformation solver, hairstyling constraint*

1 Introduction

HAIR constitutes an important part of a person's overall appearance. The same face can make quite different impressions as the person's hairstyle is varied. Therefore, to create visually convincing computer-generated humans, it is imperative to develop techniques for synthesizing realistic hair.

Synthesizing hair requires the modeling, animation, and rendering techniques. In this paper, we focus on the modeling aspect. We aim to develop an interactive technique to generate static human hairstyles in the presence of gravity, with a proper treatment of collisions.

Hair strands are very thin (40 to 80 microns in shaft diameter for medium-sized strands [20]) and are subject to deformation. Therefore, *physically-based* methods, such as finite element analysis or mass-spring simulation, can be used to model the shape of a strand in a gravitational field. A simple collection of the strands generated by the above simulation methods, however, would appear different from hair we normally see because hair strands constantly touch or collide with each other; therefore, the interactions among strands must be accounted for. Considering that there are normally about 100,000 strands of hair [23], the above task would require an enormous amount of computation if only physically-based methods were used.

The extreme opposite of the physically-based method would be to *manually* control every detail, including the effect of gravity and collisions, as well as the variations in hair strands. This would be labor-intensive and, more importantly, the result would appear unnatural unless the manual creation properly conveyed the complexity of real hair.

This paper propose a pseudophysical approach that produces realistic hairstyles from a few intuitive parameters and provides a means to edit the style in a straightforward way. Our method runs fast, thus works interactively; but it does not require the users to engage in a cumbersome manual effort. For example, the effects of gravity and collisions are automatically generated by the hair deformation solver (Section 3). Variations among strands are generated using statistical methods (Section 2) that require only a few parameters to be input by the users. As a result, our method has a short turnaround time for modeling a hairstyle.

We note that the framework presented in this paper works in a fashion that is quite simple considering the complexity and variety of the hairstyles it generates. Rather than applying *ad hoc* methods to create different styles, the framework produces various hairstyles by modifying a small number of modeling parameters. Still, the synthesized hairstyles are realistic and demonstrate a wide range of hairdos in the real world.

1.1 Related Work

In this section, we review the previous work on modeling hair geometry, computing hair deformations due to gravity and collisions, and rendering techniques that can be applied to hair.

1.1.1 Modeling Hair Geometry

To model hair by populating a number of strands, the geometry of each individual strand must eventually be modeled. Noting that adjacent hair strands tend to be alike, Watanabe and Suenaga [27] introduced a *wisp* model to generate a group of similar neighboring strands by adding variations to one key strand. Since then, different methods, such as the thin shell volume [12] and the generalized cylinder [29,30], have been used to represent the wisps. Recently, Kim and Neumann [14] proposed a multi-resolution wisp structure and user interfaces which could model a variety of hairstyles. Ward *et al.* [26] used level-of-detail representations which modeled individual strands and hair clusters with subdivision curves and surfaces, respectively.

The concept of wisp is an important basis of this work. We also represent wisps using generalized cylinders. However, unlike the previous approaches which model strands and wisps manually or represent them by parametric functions,

we introduce a *statistical wisp model*. Starting from a key strand, we construct a wisp by duplicating the key strand while constraining the wisp shape by some parameters with statistical meanings. Moreover, to generate the key strands, we introduce a Markov chain hair model by adopting the method proposed by Hertzmann *et al.* [10].

There have been other approaches that do not employ the wisp model. Hadap and Magnenat-Thalmann [8] and Yu [31] modeled hair as a flow of vector fields and calculated the strand geometries according to these fields. In simulating dynamic movement of hair, interpolation-based approaches have generally been used [1,5,16,24]: To reduce the computation required, only a relatively small number of key strands are simulated, and the results are interpolated to generate the neighboring strands. The above approaches, however, are not suitable for modeling non-uniform behavior of human hair such as clustering, which is why we take a wisp-based hair model.

Apart from the above procedural or manual modeling of hair geometries, Paris *et al.* [21] proposed a technique to capture hair geometry from multiple images, which showed a new way to generate digital hair.

1.1.2 Modeling Hair Deformations

Deformations due to gravity and collisions can be dealt with by employing physically-based models such as a simplified cantilever beam model [1,16], a mass-spring-hinge model [5,24], or a combination of a multi-body serial chain and a continuum hair model [9]. Extensions to the above models to allow for the use of the wisp structure have also been proposed [3,4,22,25]. However, those methods were targeted for *animating* hair of some limited styles, thus were not suitable for generating a variety of static hairstyles. Therefore, in the systems dedicated to modeling complicated hairstyles [14,29,30], the deformed hair shapes have been created *manually*.

In this work, we propose a new hair deformation solver which is different from the previous approaches: It is tailored to handle static hairstyles and automatically generates the shape of hair that has been deformed due to the effects of gravity and collisions. The method is based on the physical principles involved in hair deformation, but is simplified to solve only static hair shapes.

1.1.3 Rendering Hair

To render a hair strand, Kajiya and Kay's reflectance model [7,11] has been widely used; we also use this model in this work. Lately, Marschner *et al.* [19] measured the scattering from real individual hair fibers and proposed a shading model to render complicated scattering effects based on these observations. To account for self-shadowing among hair strands, LeBlanc *et al.* [15] proposed a method that uses pixel blending and shadow buffers. The deep shadow map proposed by Lokovic and Veach [18] extended the concept of a conventional shadow map to cover volumetric objects such as smoke and hair. Kim and Neumann [13] showed that the deep shadow map can be implemented efficiently on graphics hardware.

1.2 Algorithm Overview

Our algorithm for hair modeling consists of three steps—generating wisps, solving hair deformation, and constraint-based styling.

1. **Generating wisps.** A whole hair is formed by generating a collection of wisps. We determine the geometrical shape of a wisp by manipulating a representative strand and controlling the statistical properties of the surrounding member strands.
2. **Solving hair deformation.** The hair deformation solver accounts for the effects of gravity, hair elasticity, and collisions to determine the deformed shape of the wisps. This procedure is essential for relieving the user of excessive manual effort, yet it is flexible enough to handle a wide range of hairstyles.
3. **Constraint-based styling.** To model hairstyles such as braided hair, we propose a constraint-based approach to make wisps pass through a certain point or follow a certain path. This approach is effective in modeling hairstyles that involve artificial styling pieces such as hairpins or ribbons.

1.3 Paper Organization

The remainder of this paper is organized as follows: Sections 2, 3, and 4 present the statistical wisp model, the hair deformation solver, and the constraint-based styler, respectively; Section 5 describes how rendering is implemented; Section 6 summarizes experimental results; and Section 7 concludes the paper.

2 Statistical Wisp Model

It is readily observable that hair exhibits clustering behavior; a group of hair strands that are spatially close and geometrically similar is called a *wisp*. We view the task of synthesizing a hairstyle as generating a collection of wisps. This section describes how we generate the hair strands and wisps by controlling the statistical properties of hair.

2.1 Wisp Modeling

Our wisp model consists of a *master* strand and numerous *member* strands, and the overall shape of the wisp is a generalized cylinder. This section describes the wisp generation algorithm under the assumption that the geometry of the master strand is given; the method for generating the master strand is presented in Section 2.2.

We represent a strand as a Catmull-Rom spline, which is defined by and passes through a sequence of control points $\{\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_n}\}$. The spline $\mathbf{p}(s)$ is parameterized within the range $[0,1]$ so that $\mathbf{p}(0) = \mathbf{p_1}$ and $\mathbf{p}(1) = \mathbf{p_n}$ correspond to the root and tip of the strand, respectively.

Within a wisp, the degree of similarity among the strands is controlled by the length distribution, the deviation radius function, and the strand-shape fuzziness value.

- **Length distribution** $l(u)$ is a probability density function that gives the probability that a member strand will have

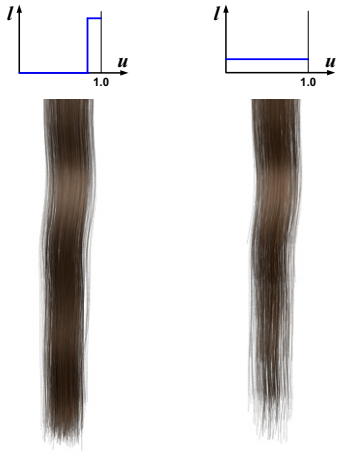


Fig. 1. Wisps generated by two different length distributions.

Fig. 2. Wisps generated by two different deviation radius functions.

Fig. 3. Wisps generated by two different fuzziness values.

length u, relative to the length of the master strand (Fig. 1).

- **Deviation radius function** $r(s)$ specifies an upper limit on the positional offset of the member strands from the master strand at the parameter value s; it controls the shape of the generalized cylinder representing the wisp (Fig. 2).
- **Strand-shape fuzziness value** σ controls the variation of the strands within a wisp. Fig. 3 shows the appearance of wisps at $\sigma = 0.1$ (low variance) and $\sigma = 1$ (high variance).

Now, the member strands are formed from the master strand by applying $l(u)$, $r(s)$ and σ. The k-th control point $\mathbf{p_k}$ of a member strand is computed by adding a displacement $\mathbf{d_k}$ to the k-th control point $\mathbf{p_k^M}$ of the master strand:

$$\mathbf{p_k} = \mathbf{p_k^M} + \mathbf{d_k}.$$

Assuming that the displacement at the root $\mathbf{d_1} = \mathbf{p_1} - \mathbf{p_1^M}$ is known, $\mathbf{d_k}$ is computed iteratively using the following equation:

$$\mathbf{d_k} = \frac{r_k}{r_{k-1}}\mathbf{d_{k-1}} + \mathbf{e}, \tag{1}$$

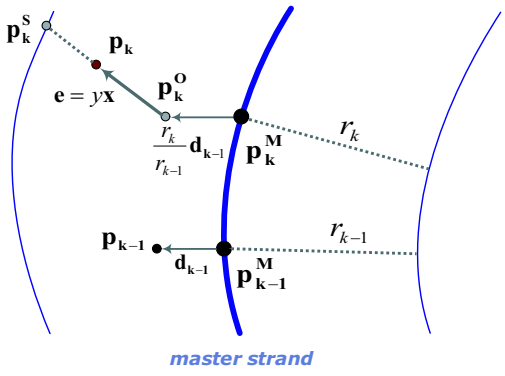


Fig. 4. Computing the point $\mathbf{p_k}$ of a member strand from the points $\mathbf{p_k^M}$ and $\mathbf{p_{k-1}^M}$ of the master strand.

where r_{k-1} and r_k are the deviation radii at $\mathbf{p_{k-1}}$ and $\mathbf{p_k}$, respectively, and $\mathbf{e}$ is a three-dimensional noise vector.

The vector $\mathbf{e}$ cannot have an arbitrary value because $\mathbf{d_k}$ must lie within the sphere of radius r_k according to the definition of the deviation radius function. Let the noise vector be expressed as $\mathbf{e} = y\bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is a unit direction vector and y is a scalar value. To determine the noise vector $\mathbf{e}$:

1. $\bar{\mathbf{x}}$ is randomly selected;
2. y is randomly chosen while $|\mathbf{d_k}| < r_k$ is satisfied.

To find y, a ray is cast originating from $\mathbf{p_k^O} = \mathbf{p_k^M} + \frac{r_k}{r_{k-1}}\mathbf{d_{k-1}}$ in the direction of $\bar{\mathbf{x}}$ until the ray intersects the sphere at a point $\mathbf{p_k^S}$. Let L be the distance from $\mathbf{p_k^O}$ to $\mathbf{p_k^S}$. A value for y is then chosen from the uniform distribution of the range $[0, \min(L, \sigma \cdot r_k)]$ as illustrated in Fig. 4. Therefore, $\mathbf{d_k}$ remains within the sphere, and the control points of the member strands are not concentrated at the boundary of the sphere.

The above procedure is repeated until the member strand attains the desired length u relative to the master strand.

2.2 Master Strand Modeling

The wisp modeling algorithm assumes that the master strands have been already given. This section presents how we actually model the geometry of a master strand.

It is generally accepted that hair strands from a single person resemble each other, which forms a unique characteristic of the person's hairstyle. In our wisp model, an effective way of ensuring that the entire hair possesses a common pattern is to enforce that all the master strands have the same pattern. Then the member strand generation algorithm (Section 2.1) extends this pattern to the entire hair.

We obtain the master strands of a common pattern by synthesizing new strands that resemble a given *prototype strand*. To synthesize a new similar strand from the prototype, we adopt the example-based Markov chain models [6,10,28], but formulate it as a Gibbs distribution to mathematically control the degree of similarity between the prototype and the synthesized strand.

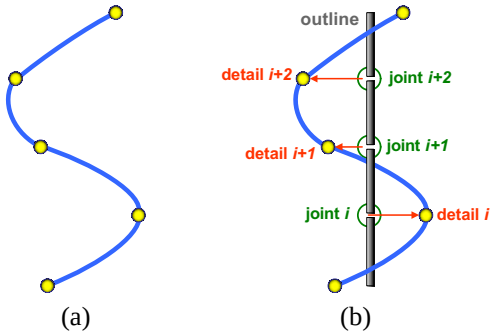


Fig. 5. Representation of the master strand: (a) the original curve, (b) the outline and details components.

2.2.1 Strand Representation

Instead of directly manipulating the control points $\{\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_n}\}$ of a master strand, we decompose them into an *outline* component, $\{\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_n}\}$, and a *details* component, $\{\delta_1, \delta_2, \cdots, \delta_n\}$, such that $\mathbf{p_i} = \mathbf{c_i} + \delta_\mathbf{i}$ $(i = 1, 2, \cdots, n)$. In Fig. 5 (b), the straight line segments at the center correspond to the outline component of the strand; they are connected with 3-DOF rotary joints, and $\mathbf{c_i}$ represents the position of the joint center. We use the segments of the equal length between the joints to simplify the calculation of deformations in Section 3. The displacements from the joint centers to the control points correspond to the details component.

The purpose of the decomposition is to separate the intrinsic geometry of the strand from the deformations applied to it by styling procedures. The subsequent deformations of the strand modify only the joint angles of the outline component, but do not modify the details component. As a consequence, the geometrical characteristics—which are encoded by the details component—can be preserved.

2.2.2 Strand Synthesis

The outline component of the strand consists of straight line segments, thus is determined straightforwardly once the segment length L_s is given. This section describes how the details component is generated from a prototype strand.

The control points of a new strand must be established so that the strand resembles the given prototype. Assuming that a hair strand can be modeled as a Markov chain, we can construct the details component by sequentially determining $\delta_\mathbf{i}$ $(i = 1, 2, \cdots, n)$ until the curve has reached the desired length while observing only the neighborhood of $\delta_\mathbf{i}$. Let $\{\delta_\mathbf{1}^*, \delta_\mathbf{2}^*, \cdots, \delta_\mathbf{m}^*\}$ be the details component of the prototype strand. To determine $\delta_\mathbf{i}$, we examine the prototype strand and find the best match $\delta_\mathbf{k}^*$ while restricting the comparison only to the *neighborhood* of $\delta_\mathbf{i}$ as illustrated in Fig. 6.[1]

The neighborhood is modeled as a window—a set of control points—around the current control point. Intuitively, the size of the window should be on the scale of the largest regular structure of the curve; otherwise, the structure may be lost. We also let the neighborhood be causal: We construct the neighborhood containing only the points preceding the current point. Therefore, in the sequence of the details component $\{\delta_\mathbf{1}, \delta_\mathbf{2}, \cdots, \delta_\mathbf{n}\}$, the neighborhood of $\delta_\mathbf{i}$ is defined by

$$\omega(\delta_\mathbf{i}) = \{\delta_{\mathbf{i-h}}, \delta_{\mathbf{i-h+1}}, \cdots, \delta_{\mathbf{i-1}}\},$$

where h is the window size.

Let $\varepsilon(\omega_i, \omega_j)$ be the Euclidean distance between the two h-sized windows $\omega(\delta_\mathbf{i})$ and $\omega(\delta_\mathbf{j})$:

$$\varepsilon(\omega_i, \omega_j) = \sqrt{\sum_{t=1}^{h} \left|\delta_{\mathbf{i-t}} - \delta_{\mathbf{j-t}}\right|^2}. \tag{2}$$

Now, the problem of determining $\delta_\mathbf{i}$ so that $\omega(\delta_\mathbf{i})$ resembles the prototype pattern can be reduced to finding a portion $\omega(\delta_\mathbf{k}^*) = \{\delta_{\mathbf{k-h}}^*, \delta_{\mathbf{k-h+1}}^*, \cdots, \delta_{\mathbf{k-1}}^*\}$ of the prototype strand such that

$$\min_k \{\varepsilon(\omega_i, \omega_k^*)\}. \tag{3}$$

Then we can deterministically specify $\delta_\mathbf{i}$ to be $\delta_\mathbf{i} = \delta_\mathbf{k}^*$.

However, the diversity of synthesized strands can be increased by stochastically selecting the details component instead of the optimal choice given by Eq. (3). We introduce the Gibbs distribution as a selection probability P_k for each possible window of the prototype strand:

$$P_k = \frac{1}{Z} \exp\left(-\frac{\varepsilon(\omega_i, \omega_k^*)}{T}\right), \tag{4}$$

where T is the temperature of the system, which can be controlled by the user, and Z is the partition function defined by $Z = \sum_k \exp\left(-\varepsilon(\omega_i, \omega_k^*)/T\right)$. The strand diversity increases as T increases, which is demonstrated in Fig. 7.

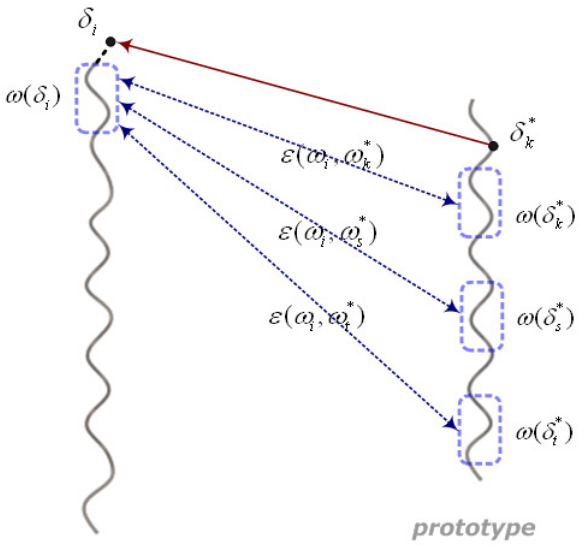
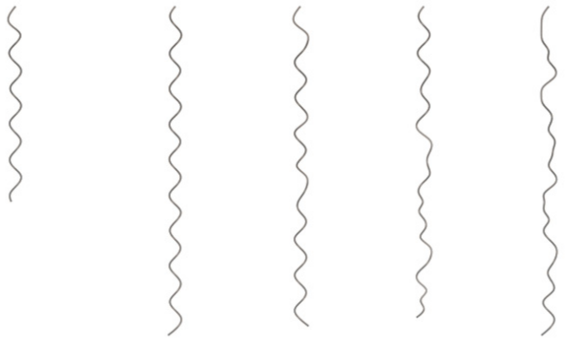


Fig. 6. Synthesizing a new strand from a prototype: To find $\delta_\mathbf{i}$, the window $\omega(\delta_\mathbf{i})$ is compared with every possible window in the prototype, then $\delta_\mathbf{k}^*$ is selected from the prototype based on Eq. (4).

[1]We adopt an approach similar to that used in "curve analogy" [10], which can be referenced for the detailed description of the curve synthesis algorithms using Markov chains.



(a) (b) (c) (d) (e)

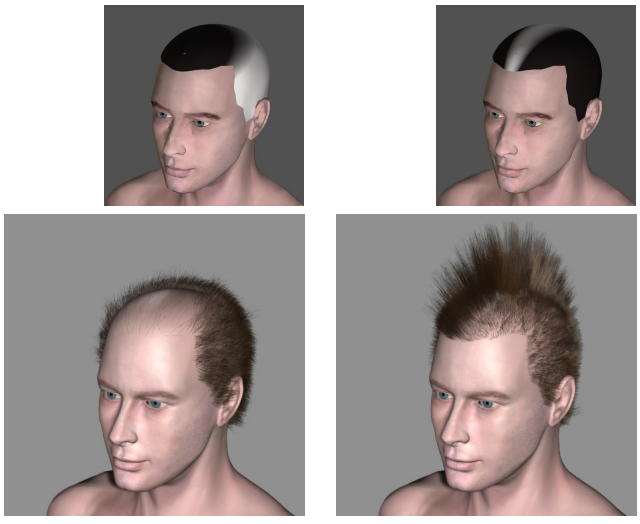
Fig. 7. The effects of T on the master strand shape: (a) the prototype strand and synthesized strands when (b) $T = 0.01$, (c) $T = 0.03$, (d) $T = 0.07$, (e) $T = 0.12$.

2.3 Global Hair Parameters

At the topmost level, the user sets the number of wisps N_w and the total number of hair strands N_s. The user then interactively constructs the density, length, and protrusion maps which define the corresponding hair properties for every point in the scalp region. The effects of the density and length maps are illustrated in Fig. 8.

Once N_w is given, the root positions of the master strands are determined by evenly distributing the number of points over the scalp region. Voronoi diagrams can then be generated to set the boundary of each wisp. Finally, using the density map and N_s, the roots of the member strands are randomly located within each Voronoi region.

Note that the length and protrusion maps are used to determine the length and protrusion direction of the *master* strands. For the member strands, these properties are determined by the algorithm presented in Section 2.1.



(a) (b)

Fig. 8. Density and length maps and their effects: (a) a density map (top)—with bright area representing high density and dark area representing low density—and its effect on hair (bottom), (b) a length map (top)—with bright area representing long hair and dark area representing short hair—and its effect on hair (bottom).

3 Hair Deformation Solver

In this section, we address the problem of determining the shape of wisps under the influence of gravity and collisions. The purpose of the hair deformation solver is to find the *joint angles* of the master strands after these factors are taken into account. Then the shape of the member strands is routinely determined from the shape of the master strand (Section 2.1).

Our hair deformation solver is based on two ideas. Firstly, it incorporates the physical properties of hair, but avoids dynamics simulation. Styling operations such as braiding assemble hair into a complex structure that is beyond the realm of simple Newtonian mechanics. Instead of simulating those situations in a *pure* physically-based way, we deal with gravity and the current styling operation as a unified force field, and solve the deformation employing pseudophysical approaches. Secondly, the deformation solver models hair as a continuum, as proposed by Hadap and Magnenat-Thalmann [9], and interprets density as a measure of collision. When the density of a certain region is above a threshold, it is regarded as a collision and the hair is forced to occupy a larger volume.

We implement the two ideas within a single framework. The deformation solver works quickly, and can be applied to a wide range of hairstyles.

3.1 Deformation Due to Styling Force Field

The *styling force field* $\Phi(\mathbf{x})$, defined in 3D space, quantitatively combines the effects of gravity and styling operations to represent the desired flow direction and intensity at each 3D point $\mathbf{x}$. The force fields rotate the joint so that the *outline segment* (Figure 5) attached to the joint is oriented along $\Phi(\mathbf{x})$. However, hair has an elastic property that resists such deformation in proportion to the bending amount and stiffness. Therefore, we determine the orientation of a segment by finding the joint angle that maximizes the following equation:

$$E = E_\Phi + \kappa E_B, \tag{5}$$

where E_Φ represents the degree to which the strand is aligned with the force field, and E_B represents the degree to which the strand is aligned with its rest position.

Let $\mathbf{q}$ be the unit direction vector of the segment. E_Φ is calculated by the following function:

$$E_\Phi = \Phi(\mathbf{x}) \cdot \mathbf{q},$$

and its value is greatest when $\mathbf{q}$ is aligned with $\Phi(\mathbf{x})$. E_B represents the (inversely proportional) amount of bending by

$$E_B = \mathbf{q_0} \cdot \mathbf{q},$$

where $\mathbf{q_0}$ is the direction of the segment when the joint angle is zero. Finally, κ is the scalar value that models the bending stiffness of hair. Fig. 9 (a) and (b) show the resulting deformation for two different values of κ.

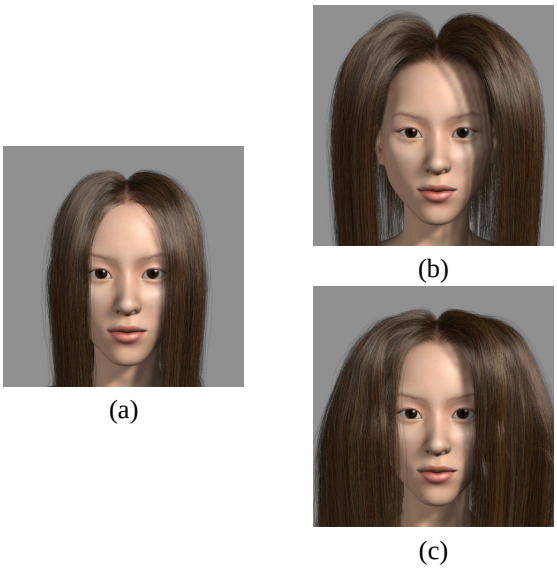


Fig. 9. Effects of bending stiffness κ and density threshold τ on hairstyles: (a) $\kappa = 0.5$, $\tau = 0.9$; (b) $\kappa = 2.5$, $\tau = 0.9$; (c) $\kappa = 0.5$, $\tau = 0.1$.

Since the optimization function in Eq. (5) is linear, it can be easily solved to find the unit vector $\mathbf{q}^*$ that maximizes it. The joint angle corresponding to $\mathbf{q}^*$ can then be calculated straightforwardly.

The above procedure determines the angle for a single joint. To determine the shape of an entire master strand, the procedure is repeated along the strand, from the root to the tip.

3.2 Deformation Due to Density Field

We now account for the hair-to-head and hair-to-hair collisions. Detecting collisions between every pair of strands would be computationally intractable. Fortunately, when we observe a hairstyle, collisions between individual strands are not noticeable. When wisps cross each other, however, it produces unmistakable artifacts. Therefore, our collision resolution method is developed to handle collisions at the wisp level.

We interpret that hair produces a density field.[2] When the procedure in Section 3.1 would locate a wisp segment in a position $\mathbf{x}$, it is first checked whether the wisp segment introduces a collision by:

$$\rho(\mathbf{x}) + \rho_\Delta(\mathbf{x}) > \tau, \tag{6}$$

where $\rho(\mathbf{x})$ is the existing density value at $\mathbf{x}$, $\rho_\Delta(\mathbf{x})$ is the increased density that would be added due to the positioning of the wisp segment, and τ is the density threshold. When the sum on the left-hand side of Eq. (6) is greater than τ, this is treated as a collision and the segment is reoriented based on the gradient of the density field. When a small value is used for τ, the overall hair volume becomes larger, as demonstrated in Fig. 9 (c).

[2]Even though the geometries of the strands are defined after the wisps are generated (Section 2), the strands are regarded as not existing yet. When the hair deformation solver starts processing strands, the density over the entire space is zero. Once the solver calculates a segment, the density corresponding to the segment is created.

3.3 Implementation Using a 3D Grid Structure

The description of the previous two sections was based on continuous fields. To implement the algorithms in a discretized space, a three-dimensional grid is constructed around the head and shoulder that includes all regions where hair may potentially be placed during the styling process. The values of the styling force and density are then stored only for points on the grid. The styling force and density at an arbitrary point in the 3D space is obtained by performing a trilinear interpolation of the values at the eight nearest grid points.

The hair deformation solver works by repeating the following two steps: (1) Deform the strands based on the fields, and (2) update the density field based on the deformation. One question remains: whether the steps should run in the breadth-first order or depth-first order. We choose the *breadth-first* order with the rationale that this handles hair-to-hair collisions better than the depth-first order does, especially because the master strands have the segments of the equal length (Section 2). We summarize the procedure for the hair deformation solver in Algorithm 1.

Algorithm 1 Hair deformation solver

```
initializeDensityField(); // 1 inside and 0 outside the head
for joint = 1 to J /* root-to-tip order */ do
  for wisp = 1 to W do
    deformByStylingForceField(); /* Eq. (5) */
    for grid points occupied by the wisp segment do
      while checkCollision() /* Eq. (6) */ do
        bend the joint by Δθ along the density gradient;
      end while
    end for
    updateDensityField(); // add the density of this segment
  end for
end for
```

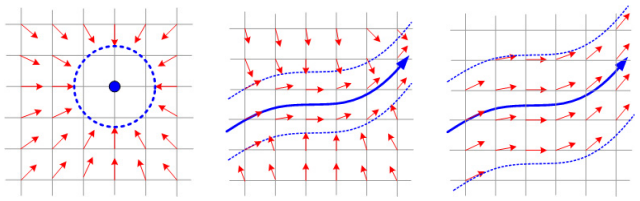
4 Constraint-Based Styler

The hair deformation solver is quite powerful in producing the styles that are based on the natural flow of hair under the gravity field. However, the method is not easily applicable to producing artificial hairstyles such as braids. To produce such types of hairstyles, we take a new approach—*styling constraints*.

A constraint causes a constraint force field $\Psi(\mathbf{x})$ to be generated over a portion of the 3D space, as shown in Fig. 10. When and only when the hair deformation solver processes the *portion being constrained*, instead of the original styling force field $\Phi(\mathbf{x})$, it uses a modified styling force field $\Phi'(\mathbf{x})$ given by

$$\Phi'(\mathbf{x}) = (1-w)\Phi(\mathbf{x}) + w\Psi(\mathbf{x}), \tag{7}$$

where the control parameter w is the weight of the constraint force field $\Psi(\mathbf{x})$ relative to the original styling force field $\Phi(\mathbf{x})$. Other than the selective superimposition of the constraint force



(a) (b) (c)

Fig. 10. Three styling constraints: (a) point constraint, (b) trajectory constraint, (c) direction constraint.

field over the original styling force field, the hair deformation solver works in the same way.

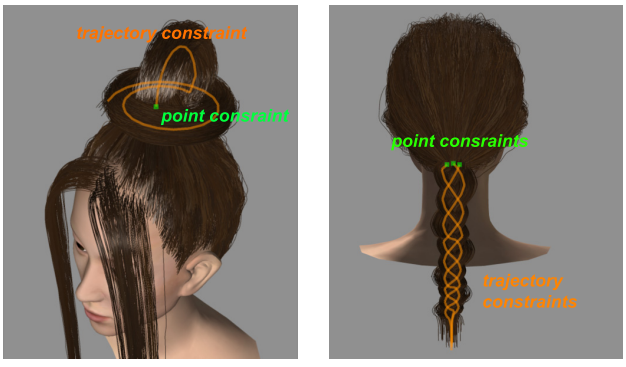
Based on the results of our experiments, we have concluded that the three types of styling constraints illustrated in Fig. 10 can be used to create most of the artificial hairstyles.

- **Point Constraint:** A point constraint is specified by an attraction point and a tolerance radius. This constraint produces vector fields toward the attraction point, as shown in Fig. 10 (a). The constraint force vector is activated only until the wisp comes within the sphere.
- **Trajectory Constraint:** A trajectory constraint is specified by a trajectory and a tolerance radius. The constraint generates unit vectors (1) toward the nearest point of the trajectory for grid points lying outside the tolerance radius, or (2) in the tangential direction of the trajectory for grid points lying inside the tolerance radius, as shown in Fig. 10 (b).
- **Direction Constraint:** A direction constraint is specified by a trajectory and an influence radius. For grid points lying within the influence radius, the constraint generates unit vectors in the tangential direction of the trajectory, as shown in Fig. 10 (c).

The functioning of the constraint-based styler can be summarized as follows:

1. Select the portion of hair, consisting of a set of wisps, upon which to apply the constraints.
2. Build a *constraint queue*, a sequence of constraints, to apply to the selected portion.

When the hair deformation solver processes those selected wisps, the constraint force field $\Psi(\mathbf{x})$ is superimposed with the styling force field $\Phi(\mathbf{x})$. The constraint-based styler activates the first constraint in the queue and processes the selected wisps until they all meet the constraint. It then processes each subsequent constraint, one at a time, in the queue. When all of the constraints have been processed, the hair deformation solver resumes processing the wisps in the usual way. Figure 11 demonstrates the use of two different constraint queues.



(a) (b)

Fig. 11. Examples of using the constraint-based styler: (a) applying a single [point + trajectory] constraint queue to a group of hair, (b) applying three [point + trajectory] constraint queues to three groups of hair, respectively. Fig. 15 (c) and (d) show the rendered images of these hairstyles.

5 Rendering

The geometric models of hair would not look natural unless they are rendered properly. Rendering quality is more important for hair than other parts of the body. This section addresses two problems: how to determine the colors of individual strands, and how to render the hair geometries.

The colors of strands taken from the same person might appear to be similar. However, closer examination reveals that no two hair strands have exactly the same color. To modulate the change of hair colors, a stochastic approach is employed. We adopt HSV color space, and model hue, saturation and value (brightness) channels as three independent probability density functions, using either uniform or Gaussian distributions. The different effects produced by adopting these two distributions are shown in Fig. 12 (b) and (c). We can also control the color variations at the wisp level, as shown in Fig. 12 (d).

In order to explicitly render individual hair strands, the Catmull-Rom splines, representing the strands (Section 2), are converted to thin ribbons using the `RiCurves` primitive in RenderMan^{TM} [2]. Then the hair shading model proposed by Kajiya and Kay [11] is used for shading each strand. Finally, the deep shadow map [18] is used to represent the self-shadowing among hair strands, which is essential for creating the volumetric appearance of hair.

(a) (b) (c) (d)

Fig. 12. Hair color corresponding to different probability distributions: (a) constant color, (b) uniform distribution, (c) Gaussian distribution, (d) variation of colors in the wisp level.

Fig. 13. Hairstyles produced by applying the statistical wisp model and the hair deformation solver. Each hairstyle has approximately 80,000 to 100,000 strands.

(a) (b) (c) (d)

Fig. 14. Synthetic hairstyles inspired by real ones: (a) a real hairstyle and (b) a corresponding synthetic one generated by applying the statistical wisp model and hair deformation solver; (c) a real updo hairstyle and (d) a synthetic imitation of it obtained by applying the constraint-based styler (images by the courtesy of http://www.hairboutique.com).

6 Results

We implemented the presented techniques on a PC with an Intel Pentium (4) 2.54GHz CPU and an NVIDIA GeForce FX 5600 GPU. To check the intermediate results during styling processes, we also implemented a renderer on a programmable graphics hardware [17].

Using our hair modeling system, both ordinary users and professional hairstylists were asked to either reproduce hairstyles from beauty magazines or to create novel hairstyles.

To help the readers understand how the whole system works, we summarize the steps taken for the hairstyling. The user starts the styling work on a given 3D polygon model of head and shoulder. In a preprocessing step, the scalp surface is specified as a polygonal surface, which defines the region where hair follicles exist. Then, the following steps are taken:

1. Construct the density, length, and protrusion maps.
2. Tune the global parameters—the number of wisps N_w and the number of strands N_s.
3. Edit the geometry of the prototype strand: The user models the prototype strand with a 3D spline curve.[3]
4. Tune the wisp parameters—the length distribution $l(u)$, deviation radius function $r(s)$, and fuzziness value σ.
5. Setup the styling force field (see Section 6.1 for the details).
6. Tune the deformation parameters—the bending stiffness κ and the density threshold τ.
7. Specify the hair color: When the user uses the Gaussian color model, he/she inputs the mean and variance of three HSV channels.

Fig. 13 shows some hairstyles created by taking the above modeling steps. Approximately 80,000 to 100,000 hair strands were used for generating each hairstyle. The computation time

[3]We also provided pre-modeled samples representing the various types of curly and wavy hair so that the user could simply select one from the samples.

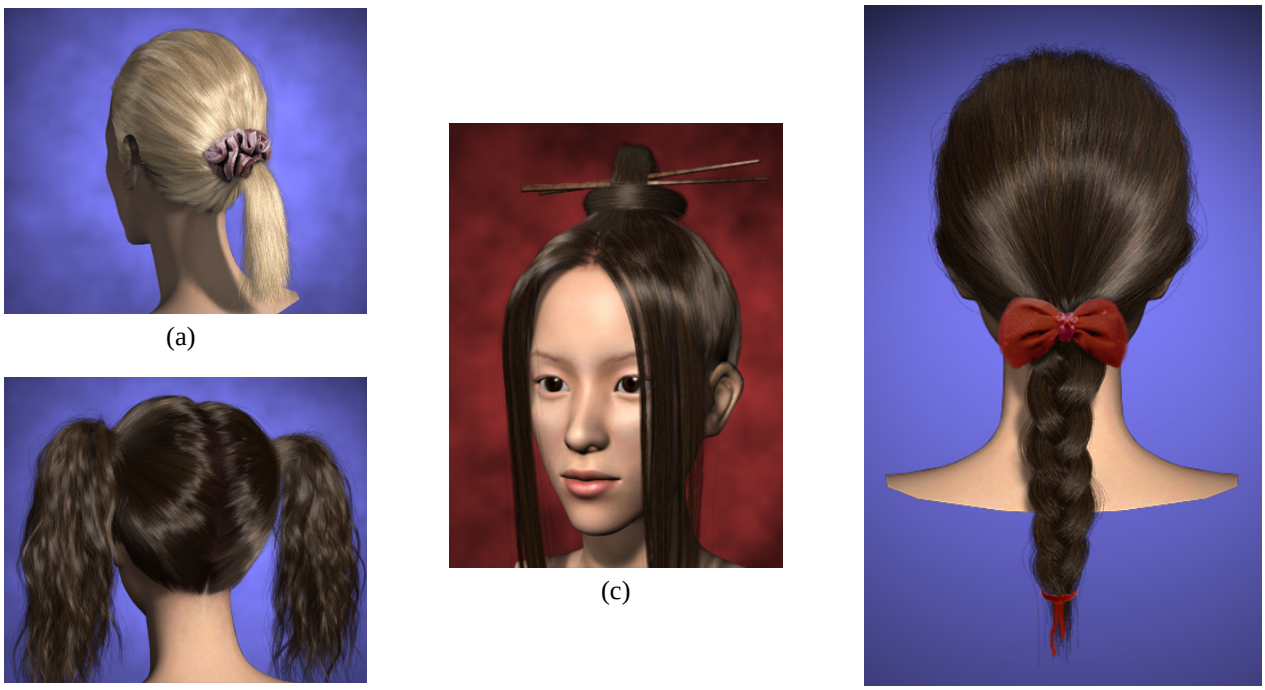


Fig. 15. Hairstyles produced by the constraint-based styler after applying (a) a single point constraint, (b) two point constraints, (c) a point constraint followed by a trajectory constraint, (d) three sets of a point constraint followed by a trajectory constraint.

greatly depended on the number of control points in a strand as well as N_w and N_s. In the initial setup, most time was spent on user interactions for preparing input to the system. However, once the input was provided, usually the result could be seen in less than 10 seconds on the hardware renderer which rendered about 20,000 strands. The accompanying video shows, step by step, how the above modeling procedure is done.

When a desired hairstyle was formed after tuning the parameters, it finally went through a software renderer, which produced the results shown in Fig. 13. The software rendering took about 30 minutes to render a hairstyle. Fig. 14 shows real hairstyles and the corresponding synthetic imitations modeled after the real ones.

6.1 Producing Styling Force Field

Currently, we construct the styling force field using a procedural approach. For convenience, we provide the users predefined force fields—e.g., gravity field, pulled-back hair field. To allow general users to fully exploit our system, a tool for editing vector fields, such as the one described in [8], should be implemented in the future.

It needs to be noted that our use of vector field is somewhat different from that in the previous work [8,31], where almost all the aspects of hairstyles are *directly* controlled by the vector field. For instance, the vector field should be carefully constructed to prevent the penetration of hair into head. The strand details are also controlled by the vector field: e.g., curly hair is generated by applying perturbations to the vector field.

On the other hand, the construction of the force field in our system is much less cumbersome since the field is used only for the rough-level control of a hairstyle. The detailed properties of hair depend on separate controls. Preventing the penetration of hair into the head is done by the collision resolution procedure (Section 3.2). Geometric details of hair strands are generated inside the wisp model (Section 2). Therefore, even though the current procedural construction is not an ultimate interface for specifying the force fields in general, it already provides enough flexibility to generate the variety of hairstyles in Fig. 13.

6.2 Hairdressing Operations

Since the five hairdressing operations—cutting, permanent waving, combing, tying, and braiding—would produce most human hairstyles, showing that our modeling system is capable of implementing these operations can give an estimation of the applicable range of the proposed technique.

- **Cutting:** It can be implemented by directly editing the length map.
- **Permanent waving:** It can be achieved by giving a prototype strand of the desired shape to the statistical wisp model.
- **Combing:** It can be implemented using a direction constraint. First, a portion of hair is selected for combing, then the desired trajectory is indicated. Finally, we specify the weight w of Eq. (7) that controls the influence of the constraint force field relative to the styling force field.
- **Tying:** The portion to tie is selected, and a point constraint is imposed so that the selected portion is attracted to the point. A value of $w = 1$ is used so that the portion is entirely affected by the constraint force field. Once the

constraint is met, the selected portion is then affected by the normal styling force field.

- **Braiding:** It can be implemented by applying a point constraint followed by a trajectory constraint to three or more groups of hair. Each group of strands is first gathered at the point specified by the point constraint. Then the trajectory constraint is applied so that the strands form braids.

Fig. 15 demonstrates several hairstyles produced by applying these operations. All hairstyles commonly involved cutting, permanent waving and combing. We additionally applied tying operations to produce the styles in Fig. 15 (a) and (b), a variation of the tying operation followed by a trajectory constraint for an updo style in Fig. 15 (c), and the braiding operation for Fig. 15 (d). In the production, the accessories, such as a ribbon, did not affect the styling, but were added just to enhance the visual realism. The accompanying video shows the user interface for hairdressing operations.

7 Conclusion

We have presented a technique for interactively modeling human hair. A variety of hairstyles can be generated by modifying a small number of wisp parameters while the effect of gravity is automatically solved by the hair deformation solver. The framework is quite versatile to be capable of realizing familiar but non-trivial styling operations such as permanent waving and braiding in a straightforward manner. The technique provides improvements over previous methods in the reality of the result and the range of hairstyles it can produce.

The speedup, realism, and diversity are possible due to the combination of the statistical wisp model, the hair deformation solver, the constraint-based styler, and the stochastic hair color model.

Firstly, the statistical wisp model has relieved us from modeling each hair strand manually or by employing parametric functions. We have introduced an example-based Markov chain model to synthesize a master strand from a given prototype curve. We duplicate a master strand to neighboring member strands by creating statistically-controlled variations.

Secondly, we have proposed a solver to compute the deformations due to the gravity and collisions. By tailoring the solver to *static* hairstyles, we could make it work fast.

Thirdly, some hairstyles, such as braids or updo styles, require tedious hairdressing operations, and have been difficult to synthesize. The constraint-based styler provides us an effective way of producing such styles. Even though there are only three types of constraints, the combinations of them can produce a wide range of styles, as shown in Figure 15.

Finally, by using probability density functions to represent the randomness of hair color, we have obtained more realistic results even with the traditional Kajiya and Kay shading model. We plan to explore using a more sophisticated shading model proposed by Marschner *et al.* [19], combined with the stochastic color model.

Limitations and Future Work: The dynamic movement of hair is not considered in this work, though the technique is still useful for many applications. Hair, especially short hair, generally moves very little unless there is a strong wind or the character makes a sudden movement; as a result, the technique can be used in some limited cases of character animation. As future work, we are developing hair simulation technique based on the wisp models presented in this paper. Another limitation of this work is the difficulty of modeling hairstyles where the wisps are closely packed so that there are too many collisions among them. To solve this problem, we could use a multi-resolution approach such as the "wisp tree" [3,14].

Acknowledgments

This research was supported by the Ministry of Information and Communication, Republic of Korea. This work was also partially supported by the Automation and Systems Research Institute at Seoul National University and the Brain Korea 21 Project. The authors would like to thank Prof. Il Dong Yun for his inspiration and guidance at the initial stage of this research.

References

[1] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, volume 26, pages 111–120, July 1992.

[2] A. A. Apodaca and L. Gritz. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, 1999.

[3] F. Bertails, T. Kim, M.-P. Cani, and U. Neumann. Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 207–213, 2003.

[4] J. T. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 73–80, July 2002.

[5] A. Daldegan, N. M. Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum (Eurographics '93)*, 12(3):211–221, 1993.

[6] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038, Sept. 1999.

[7] D. B. Goldman. Fake fur rendering. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 127–134, Aug. 1997.

[8] S. Hadap and N. Magnenat-Thalmann. Interactive hair styler based on fluid flow. In *Computer Animation and Simulation 2000*, pages 87–99, Aug. 2000.

[9] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum (Eurographics 2001)*, 20(3):329–338, 2001.

[10] A. Hertzmann, N. Oliver, B. Curless, and S. M. Seitz. Curve analogies. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 233–246, June 2002.

[11] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, volume 23, pages 271–280, July 1989.

[12] T. Kim and U. Neumann. A thin shell volume for modeling human hair. In *Computer Animation 2000*, pages 104–111, May 2000.

[13] T. Kim and U. Neumann. Opacity shadow maps. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, pages 177–182, June 2001.

[14] T. Kim and U. Neumann. Interactive multiresolution hair modeling and editing. *ACM Transactions on Graphics (Siggraph 2002)*, 21(3):620–629, July 2002.

[15] A. LeBlanc, R. Turner, and D. Thalmann. Rendering hair using pixel blending and shadow buffer. *Journal of Visualization and Computer Animation*, 2:92–97, 1991.

[16] D. Lee and H. Ko. Natural hairstyle modeling and animation. *Graphical Models*, 63(2):67–85, Mar. 2001.
[17] E. Lindholm, M. J. Kilgard, and H. Moreton. A user-programmable vertex engine. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 149–158, Aug. 2001.
[18] T. Lokovic and E. Veach. Deep shadow maps. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 385–392, July 2000.
[19] S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. *ACM Transactions on Graphics (Siggraph 2003)*, 22(3):780–791, July 2003.
[20] R. R. Ogle and M. J. Fox. *Atlas of Human Hair: Microscopic Characteristics*. CRC Press, 1999.
[21] S. Paris, H. M. Briceño, and F. X. Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics (Siggraph 2004)*, 23(3):712–719, Aug. 2004.
[22] E. Plante, M. Cani, and P. Poulin. Capturing the complexity of hair motion. *Graphical Models*, 64(1):40–58, Jan. 2002.
[23] C. R. Robbins. *Chemical and Physical Behavior of Human Hair*. Springer-Verlag, 4th edition, Dec. 2000.
[24] R. E. Rosenblum, W. E. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: modeling, rendering and animation. *The Journal of Visualization and Computer Animation*, 2:141–148, 1991.
[25] K. Ward and M. C. Lin. Adaptive grouping and subdivision for simulating hair dynamics. In *Pacific Conference on Computer Graphics and Applications*, 2003.
[26] K. Ward, M. C. Lin, J. Lee, S. Fisher, and D. Macri. Modeling hair using level-of-detail representations. In *Proc. of Computer Animation and Social Agents*, 2003.
[27] Y. Watanabe and Y. Suenaga. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics & Applications*, 12(1):47–53, Jan. 1992.
[28] L. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, pages 479–488, July 2000.
[29] Z. Xu and X. D. Yang. V-hairstudio: an interactive tool for hair design. *IEEE Computer Graphics & Applications*, 21(3):36–42, 2001.
[30] X. D. Yang, Z. Xu, J. Yang, and T. Wang. The cluster hair model. *Graphical Models*, 62(2):85–103, Mar. 2000.
[31] Y. Yu. Modeling realistic virtual hairstyles. In *9th Pacific Conference on Computer Graphics and Applications*, pages 295–304, Oct. 2001.