A Parameter-Based Hairstyler

Byoungwon Choe Hyeong-Seok Ko

Graphics & Media Lab.
Seoul National University

패러미터 기반 머리카락 모델링 기법

최병원 고형석

서울대학교 그래픽스 및 미디어 연구실

Abstract

This paper presents an interactive technique that produces static human hairstyles by generating individual hair strands of the desired shape and color, subject to the presence of gravity and collisions. A variety of human hairstyles can be generated by supplying a few parameters to the modeling core, which consists of the wisp generator and hair deformation solver. Wisps are generated employing statistical approaches. As for hair deformation, a method that is not physically-based is proposed to efficiently account for the effects of gravity and collisions. The technique produces various hairstyles much faster than previously proposed methods, and the styles generated by this technique are remarkably realistic.

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, if we are to create visually convincing computer-generated humans, developing techniques for synthesizing realistic hair is imperative.

Synthesizing hair requires techniques for modeling, animating, and rendering. However, we focus on the modeling aspect in this paper. We aim to develop an interactive technique that enables the design of static human hairstyles in the presence of gravity, with a proper treatment of collisions.

This paper presents a technique that produces realistic hairstyles from a few input parameters. The styles generated by this technique are quite realistic and span a significantly wider range of human hairstyles than those produced by previously proposed methods. Our method runs fast thus works interactively, but it does not require the user to engage in cumbersome manual effort. For example, the effects of gravity and collisions are automatically generated by the hair deformation solver, which is presented in Section 3. Variations among strands are generated using statistical methods that require only a few parameters to be input by the user. Thus, our method has a short turnaround time for modeling a hairstyle.[1]

We note that the framework presented in this paper works in a fashion that is surprisingly 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 supplying a small number of parameters to the modeling core, which acts as a "black box."

1.1 Related Work

To model hair by populating a number of strands, the geometry of each individual strand must eventually be modeled. Noting that adjacent strands of hair tend to be alike, Watanabe and Suenaga [19] introduced the wisp model to generate a group of similar neighboring strands by adding variations to one key strand. Since then, different geometries such as thin shell volume [12] and generalized cylinder [22, 21, 13] have been used to represent wisps. Kim and Neumann [13] proposed a multi-resolution wisp structure and user interfaces which could model a variety of hairstyles. The concept of wisp is an important basis of this work. We use wisps of generalized cylindrical shapes. As for the key strands, we model a strand as a Markov Process.

[1] The turnaround time includes the time for hardware shading; however, it does not include the time for running a software renderer, which is needed to produce photo-realistic results.

Given a prototype strand geometry, we can then generate strands of similar shape by adopting the method proposed by Hertzmann *et al.* [10] and texture synthesis algorithms proposed by [7, 20].

There have been other approaches that do not employ the wisp model. Hadap and Magnenat-Thalmann [8] and Yu [23] 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 [17, 1, 6, 14]: 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.

Deformations due to gravity and collisions can be dealt with by employing physically-based models such as a simplified cantilever beam model [1, 14], a mass-spring-hinge model [17, 6], 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 [4, 16, 3, 18]. However, those methods were targeted for *animating* hair of some limited styles, thus were not suitable for generating a variety of static hairstyles. The techniques proposed for modeling static hairstyles provide various interfaces for manually creating the deformed hair shapes [22, 21, 13]. The method we develop in this work, the hair deformation solver, is different from the above 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 not physically-based, although it is based on the physical principles involved in hair deformation.

1.2 System Overview

The user specifies a set of parameters that are input to the modeling core, which then produces the specified hairstyle. Because each parameter has a clear intuitive meaning, the displayed result provides direct feedback to the user that guides the subsequent modification of parameter values to achieve the intended style. The modeling core consists of the wisp generator and hair deformation solver. In this section, we briefly describe what functions are performed by each component; the sections that follow explain how each of these components is implemented.

- **Wisp Generator:** In this work, a hairstyle is formed by generating a collection of wisps. Therefore, generating wisps of the desired shape is a fundamental capability required of the modeling core. The wisp generator determines the geometrical shape of a wisp by manipulating a representative strand and controlling the statistical properties of the surrounding member strands.

- **Hair Deformation Solver:** The hair deformation solver accounts for the effects of gravity, hair elasticity, and collisions to determine the deformed shape of the wisps. This component is essential for relieving the user of excessive manual effort, yet it is flexible enough to allow our styling technique to produce a wide range of human hairstyles.

1.3 Paper Organization

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

2 Wisp Generator

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; therefore, the method for generating wisps greatly affects the quality of our hairstyling technique. This section describes how we generate 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. It is generally accepted that hair strands from a single person resemble each other, thus forming a unique characteristic of the person. An effective way of ensuring that an entire head of hair possesses a common pattern is to enforce that pattern at the master strand generation stage, and extends this pattern to the remainder of the hair.

We obtain strands of a common pattern by developing a method to generate strands that resemble a given prototype strand. The procedure of generating master strands is adopted from the work of Hertzmann *et al.* [10], with several necessary modifications. The detailed algorithm is presented in [5].

When a wisp is to be deformed in subsequent styling processes, only the master strand is deformed; the shape of the

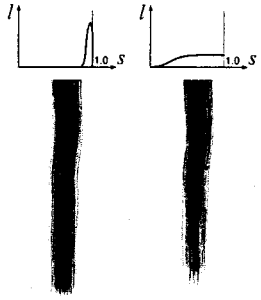


Figure 1: Wisps generated by two different length distributions.

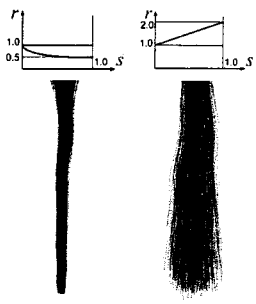


Figure 2: Wisps generated by two different deviation radius functions.

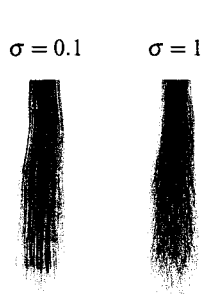


Figure 3: Wisps generated by two different fuzziness values.

member strands are routinely determined from the shape of the master strand by the method described in this section.

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 function, the deviation radius function, and the strand-shape fuzziness value. The length distribution $l(u)$ is a probability density function that gives the probability that a member strand will have length u, relative to the length of the master strand. Figure 1 illustrates examples of wisps generated from the two different length distribution functions shown. The 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. Figure 2 demonstrates how the shape of a wisp is affected by the deviation radius function. The strand-shape fuzziness value σ controls the variation of the strands within a wisp. Figure 3 shows the appearance of wisps at $\sigma = 0.1$ (high resemblance) and $\sigma = 1$ (low resemblance).

We can now describe how the member strands are formed from the master strand by applying the functions $l(u)$ and $r(s)$ and the parameter σ. The k-th control point $\mathbf{p_k}$ of a member strand is given 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, $\mathbf{d_1}$, is known (which will be addressed in Section 2.2), $\mathbf{d_k}$ is computed iteratively using the following equation:

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

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 3D noise vector. The vector $\mathbf{e}$ cannot be arbitrary because $\mathbf{d_k}$ must lie within a sphere of radius r_k. 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 a value for the noise vector $\mathbf{e}$, first a unit direction vector $\bar{\mathbf{x}}$ is randomly selected. Then, draw a ray originating at $p_k^o = \mathbf{p_k^M} + \frac{r_k}{r_{k-1}}\mathbf{d_{k-1}}$ with direction $\bar{\mathbf{x}}$, find the point p_k^s where the ray intersects the sphere, and let L be that the distance from p_k^o to p_k^s. A value for y is then chosen from the uniform distribution $[0, \min(L, \sigma)]$ as illustrated in Figure 4, thus insuring that $\mathbf{d_k}$ remains within the sphere and at the same time that 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.

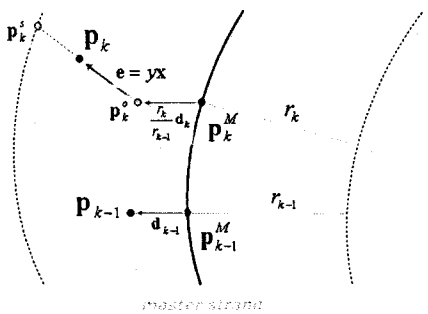


Figure 4: Computing the point $\mathbf{p_k}$ of a member strand from the master strand.

2.2 Global Control

In addition to the low-level parameters defined above in the procedures for generating wisps and master strands, there

are also some global parameters that the user must define. At the topmost level, the user must set the number N_w of wisps and the total number N_s of hair strands. The user must then interactively construct the density, length, and protrusion maps, which define the corresponding hair properties for every point in the scalp region.

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

3 Hair Deformation Solver

The wisp generation algorithm presented in the previous section assumed that the outline components of the master strands are straight. 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 determine the joint angles of the master strand segments after these factors are taken into account.

Our hair deformation solver is based on two ideas. Firstly, it avoids physically-based 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 physically-based way, therefore, we deal with gravity and the current styling operation as a unified force field around the facial region. Secondly, the deformation solver models hair as a continuum 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.

The above two ideas allow us to account for the effects of both gravity and collisions within a single framework. The deformation solver works surprisingly 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 over 3D space, is a non-physical vector that quantitatively combines the effects of gravity and styling operations to represent the desired flow direction and intensity at each 3D point $\mathbf{x}$.

The styling force field described above will create a tendency to bend the joint in such a way that the hair strand segment under consideration would become oriented along

(a) (b) (c)

Figure 5: 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) $\kappa = 0.5$, $\tau = 0.1$.

$\Phi(\mathbf{x})$. However, hair has an elastic property that resists such a deformation in proportion to the bending amount and stiffness. Therefore, we determine the orientation of a particular segment of a master strand, represented by the unit direction vector $\mathbf{q}$, by finding the joint angle that maximizes the following equation:

$$E = E_\Phi + \kappa E_B, \tag{2}$$

where E_Φ represents the degree to which the strand is aligned with the force field at that point, and E_B represents the degree to which the strand is aligned with its unbent position. 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 amount of bending with the formula:

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

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. Figure 5 (a) and (b) show the resulting deformation for two different values of κ. Because the functional in Equation (2) is linear, it can be solved directly 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

The above procedure does not account for the hair-to-head and hair-to-hair collisions. This section presents our collision handling algorithm, which is based on the density field concept.

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 to a position $\mathbf{x}$, it is first checked if the following is true:

$$\rho(\mathbf{x}) + \rho_{\Delta}(\mathbf{x}) > \tau, \tag{3}$$

where $\rho(\mathbf{x})$ is the existing density value at $\mathbf{x}$, $\rho_{\Delta}(\mathbf{x})$ is the change in 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 Equation (3) 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 Figure 5c.

3.3 Implementation Using a 3D Grid Structure

The description of the previous two sections was based on continuous fields; however, their computer implementation is carried out in a discretized space. To this end, a three-dimensional grid is constructed around the head 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 tri-linear 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 loop should run in the breadth-first order or depth-first order. We choose to perform the updates in breadth-first order with the rationale that this handles hair-to-hair collisions better than depth-first order does, especially because the master strands are modeled with segments of *equal* length, as mentioned in Section 2. We summarize the procedure for the hair deformation solver in Algorithm 1.

Algorithm 1 Hair deformation solver

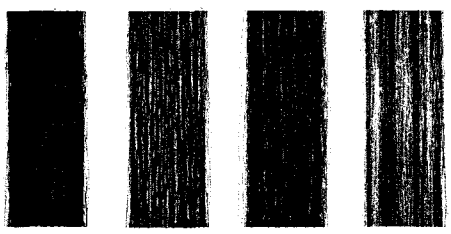
```
initializeDensityField(); // 1 inside and 0 outside the head
for joint = 1 to J /* root-to-tip order */ do
  for wisp = 1 to W do
    deformByStylingForceField(); /* Eq. (2) */
    for grid points occupied by the current wisp segment do
      while checkCollision() /* Eq. (3) */ do
        bend the joint by Δθ along the density gradient;
      end while
    end for
    updateDensityField(); // add the density of this segment
  end for
end for
```

4 Rendering

The geometric models of human hair produced by the methods described in the previous sections would not appear impressive unless they are rendered properly. Rendering quality is more important for hair than for other parts of the body. This section addresses two problems: how to determine the color of individual strands, and how to render the result.

Initially, the color of strands taken from the same person may appear to be similar. However, closer examination reveals that no two hair strands actually have exactly the same color. To modulate the similarity and variety of the hair colors produced, a stochastic approach is employed. We adopt HSV color system. We model hue, saturation, and value (brightness) of a strand as three independent probability density functions, using either a Gaussian or a uniform distribution. The different effects produced by adopting each of these two distributions are shown in Figure 6 (b) and (c). We can also control the color variations at the wisp level, as shown in Figure 6d.

In order to explicitly render individual hair strands, the Catmull-Rom splines used to represent them are converted to thin ribbons using the `RiCurves` primitive in RenderMan^{TM} [2]. The hair shading model proposed by Kajiya and Kay [11] is used for shading each strand. The deep shadow map proposed by [15] is used to represent the self-shadowing inside the hair volume, which is essential for creating the volumetric look of hair.



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

Figure 6: Hair color corresponding to different distributions: (a) constant color, (b) uniform distribution, (c) Gaussian distribution, (d) variation of colors in the wisp level.

[2] Even though the data to encode the inter-wisp and intra-wisp variations are precomputed in the wisp generator, we regard that the strands do not actually exist yet. Therefore, when the hair deformation solver starts processing strands, the density over the entire space is zero. As the solver creates wisp segments, the density develops in root-to-tip order.

Figure 7: Hairstyles produced by our parameter-based hairstyler.

5 Results

We implemented the above techniques on a PC with an Intel Pentium (4) 2.54GHz CPU and an NVIDIA GeForce FX 5600 GPU. Using our hair modeling system, the users were asked to either re-create styles from beauty magazines or to create novel hairstyles. Figure 7 shows some hairstyles created using the modeling system. Approximately 100,000 hair strands were used for each hairstyle. The styling was done progressively in an iterative fashion. Once a set of parameter values were entered by the user, it took only a few seconds to produce and display the intermediate result. Table 1 summarizes the modeling parameters used in this paper, together with their meaning.

param.	meaning
N_s	total number of strands
N_w	number of wisps
$l(u)$	length distribution within a wisp
$r(s)$	deviation radius of the generalized cylinder
σ	strand shape fuzziness
Φ	styling force field
κ	bending stiffness of a hair strand
τ	density threshold

Table 1: Summary of parameters used in this paper.

6 Conclusion

In this paper, we have presented a technique to model human hair in a gravity field. A variety of hairstyles can be generated by supplying a few parameters to the *modeling core* consisting of the wisp generator and hair deformation solver. The technique is very efficient, thus allows the styling work to be done interactively, iteratively viewing and refining the results. In addition, the technique provides a remarkable improvement over previous methods in the realism of the result.

Although the dynamic movement of hair is not considered in this work, 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.

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, and U. Neumann. Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. In *ACM SIGGRAPH Symposium on Computer Animation*, 2003.

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

[5] B. Choe. *Statistical Approaches for Synthesizing Realistic Face and Hair*. PhD thesis, Seoul National University, Feb. 2004.

[6] A. Daldegan, N. M. Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating and rendering hair. 12(3):211–221, 1993.

[7] 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.

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

[14] D. Lee and H. Ko. Natural hairstyle modeling and animation. *Graphical Models*, 63(2):67–85, Mar. 2001.

[15] 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.

[16] E. Plante, M. Cani, and P. Poulin. Capturing the complexity of hair motion. *Graphical Models*, 64(1):40–58, Jan. 2002.

[17] 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.

[18] K. Ward and M. C. Lin. Adaptive grouping and subdivision for simulating hair dynamics. In *Pacific Conference on Computer Graphics and Applications*, 2003.

[19] Y. Watanabe and Y. Suenaga. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics & Applications*, 12(1):47–53, Jan. 1992.

[20] 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.

[21] Z. Xu and X. D. Yang. V-hairstudio: an interactive tool for hair design. *IEEE Computer Graphics & Applications*, 21(3):36–42, 2001.

[22] X. D. Yang, Z. Xu, J. Yang, and T. Wang. The cluster hair model. *Graphical Models*, 62(2):85–103, Mar. 2000.

[23] Y. Yu. Modeling realistic virtual hairstyles. In *9th Pacific Conference on Computer Graphics and Applications*, pages 295–304, Oct. 2001.