

Sketching Hairstyles

Hongbo Fu Yichen Wei Chiew-Lan Tai Long Quan

The Hong Kong University of Science and Technology

Abstract

This paper presents an intuitive sketching interface for interactive hairstyle design, made possible by an efficient numerical updating scheme. The user portrays the global shape of a desired hairstyle through a few 3D style curves that are manipulated by interactively sketching freeform strokes. Our approach is based on a vector field representation that solves a sparse linear system with the style curves acting as boundary constraints. The key observation is that the specific sparseness pattern of the linear system enables an efficient incremental numerical updating scheme. This gives rise to a sketching interface that provides interactive visual feedback to the user. Interesting hairstyles can be easily created in minutes.

1. Introduction

Realistic looking hair is an important feature of virtual characters which appear in many applications, such as movies and games. While significant progress has been made on hair simulation [BAC*06] and rendering (see [MM06] and references therein), hair modeling still remains a difficult problem. This is due to the huge number of individual hair curves on a human head (typically more than 100K) and the large variance of hairstyles.

The key contribution of this paper is a hairstyle design system equipped with a sketching interface and a fast vector field solver. The user draws freeform strokes to create and edit a few style curves which depict the global shape of the desired hairstyle. The hairstyle is generated by following the flow lines in a vector field, which is transparent to the user. The vector field is formulated as the solution of a linear system $\mathbf{Ax} = \mathbf{b}$ with the style curves acting as boundary constraints. Despite the high sparsity of \mathbf{A} , directly solving the system is still too slow (more than twenty seconds for 50K variables) for user interaction. Instead, we observe that modifying the style curves induces only the changes of \mathbf{b} and the diagonal elements of \mathbf{A} . Once initialized, the linear system can be efficiently re-solved incrementally due to the special pattern, usually taking a few seconds.

The combination of the sketching interface and the efficient vector field solver gives rise to a user-friendly system. User continuously draws strokes to modify the hairstyle, responding to the interactive feedback, until satisfied. Inter-

esting hairstyles can be easily created (see Figure 1 for an example).

2. Related Work

A variety of hair modeling techniques have been proposed (see the latest survey in [WBK*07]). We review the work mostly related to ours.

Direct Hair Modeling. Many previous interactive hair modeling techniques directly manipulate the geometry of hair curves, or a group of hair curves, called a hair cluster [GW97, KN02, Mal05]. Modeling a complete hair model with such techniques could be tedious and time-consuming (usually several hours) since hundreds of hair clusters have to be created manually.

Vector Field-based Hair Modeling. Vector field-based techniques can effectively reduce manual work by automatically tracking the hair curve flow in a vector field. The idea was first explored by Hadap and Magnenat-Thalmann [HMT00]. Yu [Yu01] extended the idea by introducing more vector field primitives to create more complex hairstyles. Major limitation of these methods is that the global vector field is continuously represented as the superimposition of many local vector fields generated by those primitives. When the vector field is changed, several minutes are needed to re-evaluate the vector field and re-generate a hair model. The high computational cost makes user interaction inconvenient. Moreover, the vector field is modified via positioning and rotating primitives in space; it is sometimes not very intuitive that how a primitive will affect the

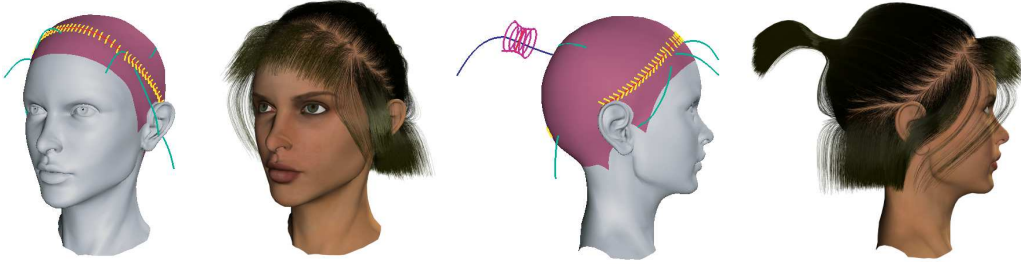


Figure 1: A realistic hairstyle created using our system in **five minutes**. The user is allowed to design interesting hairstyles by intuitively sketching three types of style primitives: streaming curve, dividing curve and ponytail.

hairstyle [Yu01]. Instead of using a single vector field, Choe et al. [CK05] proposed to use a set of styling vector fields and apply individual vector fields each time to generate more complex hairstyles, such as braid hair. However, their styling vector fields are produced using a procedural approach, not allowing users to fully design hairstyles.

Sketching Interface for Hair Modeling. Sketching interface for 3D design has been proved intuitive [ZHH96, IMT99]. Mao et al. [MKIA04] were the first to apply sketching to hair modeling, but their approach only assumes symmetric smooth hairstyles. The recent approach of Malik [Mal05] allows user to draw freeform strokes to mimic various hairstyling operations on individual hair clusters. Since the user directly manipulates the hair geometry and the influence of the editing operation is local, it is not easy to design a globally complex hair style. Wither et al. [WBC07] proposed a sketch-based interface for controlling a physically-based hairstyle generator.

Image-based Hair Modeling. Recently, researchers have shown that automatic hair geometry reconstruction based on computer vision techniques is possible [PoS04, WOQS05]. Typically multiple images are captured about the subject head, and dense hair curve geometry is recovered from hair's photometric or stereoscopic properties. While current results are impressive, the use of these approaches are limited by data capturing conditions.

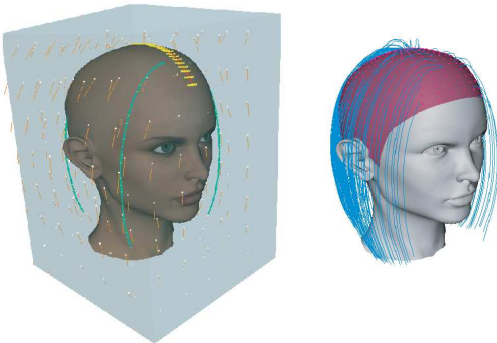


Figure 2: **Left:** bounding volume, vector field and style curves. **Right:** hair curves generated from scalp.

3. System Overview

The system consists of four components (see Figure 2): a head mesh, a vector field defined in the bounding volume, a set of style curves, and a resulting hairstyle consisting of tens of thousands of hair curves. To design a specific hairstyle, user first sketches a few style curves depicting the global shape. These style curves are created and modified via drawing freeform strokes. A discrete vector field is defined in a 3D uniform grid within the bounding volume of the head. It is formulated as the solution of a linear system. Different boundary constraints are derived from style curves for the vector field. For example, a stream curve (cyan curves in Figure 2) causes its neighboring grid points to have their directional vectors set along its tangent, and a dividing curve (red curve in Figure 2) causes the neighboring grid points on its two sides to assume roughly opposite directions (indicated as short yellow lines).

Once solved, the vector field automatically generates a hairstyle. Each hair curve starts from a root point on the scalp and grows along the flow directions in the vector field. Equipped with an efficient incremental solver for the linear system, our system allows the user to modify the style curves, re-solve the linear system and generate the new hairstyle in several seconds. The interactive visual feedback greatly facilitates the design process.

4. Fast Vector Field Computation

This section introduces the linear system which produces a vector field for hair growth. We use a Laplacian system as a field interpolator, given the boundary constraints derived from the style curves. A fast solver based on incremental Cholesky factorization is presented.

4.1. Laplacian System as Field Interpolator

In recent years, the Laplace operator has been extensively adopted in mesh editing due to its ability to produce smooth deformation (see [Sor06] and references therein) and we adopt a similar formulation. For each vertex \mathbf{v}_i of the grid, the discrete Laplace operator is defined as

$$\Delta(\mathbf{t}_i) = \sum_{j \in N(i)} \frac{1}{N(i)} (\mathbf{t}_j - \mathbf{t}_i),$$

where \mathbf{t}_i is a directional vector defined at \mathbf{v}_i and $N(i)$ is the index set of the 1-ring neighboring vertices of \mathbf{v}_i .

We formulate the problem of field interpolation as a minimization problem with cost function,

$$E(\mathbf{t}_1, \dots, \mathbf{t}_n) = \sum_{i=1}^n \|\Delta(\mathbf{t}_i)\|^2 + \omega^2 \sum_{i \in C} \|\mathbf{t}_i - \mathbf{c}_i\|^2, \quad (1)$$

where $C = \{k_1, \dots, k_m\}$ is the index set of boundary constraints, and ω is the weight of the soft constraints ($\omega = 100$ in our implementation). It is well known that the above minimization is equivalent to solving the following linear system in a least-squares sense

$$\mathbf{A}\mathbf{t}^{(x)} = \begin{pmatrix} \mathbf{D} \\ \mathbf{W} \end{pmatrix} \mathbf{t}^{(x)} = \begin{pmatrix} \mathbf{0} \\ \omega \mathbf{c}^{(x)} \end{pmatrix} = \mathbf{b}^{(x)}, \quad (2)$$

where matrix \mathbf{D} is an $n \times n$ matrix with the entries from the discrete Laplacian, $\mathbf{W} = (w_{ij})_{m \times n}$, and w_{ij} is ω if $k_i = j$, and 0 otherwise. The column vectors $\mathbf{t}^{(x)}$ and $\mathbf{c}^{(x)}$ contain the x -component of \mathbf{t}_i and \mathbf{c}_i , respectively. Similar systems are defined for y and z components. Solving Equation (2) in the least-squares sense leads to a system of normal equations:

$$\mathbf{A}^T \mathbf{A} \mathbf{t}^{(x)} = (\mathbf{D}^T \mathbf{D} + \mathbf{W}^T \mathbf{W}) \mathbf{t}^{(x)} = \mathbf{A}^T \mathbf{b}^{(x)}. \quad (3)$$

Note that $\mathbf{W}^T \mathbf{W}$ is a diagonal matrix.

4.2. Incremental Cholesky Factorization

When the user changes the style curves, the set of boundary constraints is updated (see details in Section 5). This requires $\mathbf{W}^T \mathbf{W}$ to be updated, the right hand side of Equation 3 to be changed ($\mathbf{D}^T \mathbf{D}$ remains unchanged), and the system to be re-solved. Although the system matrix is very sparse, solving the system with $50K$ unknowns still takes more than twenty seconds.

A key observation is that changing the boundary constraints (via modifying the style curves) only affects the diagonal elements of $\mathbf{W}^T \mathbf{W}$. Specifically, the number of affected elements is $|\bar{C} - C| + |C - \bar{C}|$, where C and \bar{C} are the old and new sets of boundary constraints, respectively. Consequently, for such special modifications to the system matrix, solving the normal equations by modifying the existing Cholesky factorization of $\mathbf{A}^T \mathbf{A}$ is much more efficient than solving the system from scratch [DH99]. Efficient modification of sparse Cholesky factorization is only possible for some special cases. Specifically, given a sparse positive definite matrix $\mathbf{A}^T \mathbf{A}$ and its associated Cholesky factorization, its modification is efficient only when $\mathbf{A}^T \mathbf{A}$ goes to $\mathbf{A}^T \mathbf{A} + \mathbf{R}^T \mathbf{R}$ (called an *update*) or $\mathbf{A}^T \mathbf{A} - \mathbf{R}^T \mathbf{R}$ (called a *downdate*), where \mathbf{R} is an arbitrary matrix [DH99].

We adapt the incremental Cholesky factorization. First, we perform a general sparse Cholesky factorization

$$\mathbf{L}\mathbf{L}^T = \mathbf{A}^T \mathbf{A} = \mathbf{D}^T \mathbf{D} + \mathbf{W}^T \mathbf{W},$$

which is pre-computed only once. Given the new boundary constraints set \bar{C} , similar to the definition of \mathbf{W} , we let \mathbf{W}_+

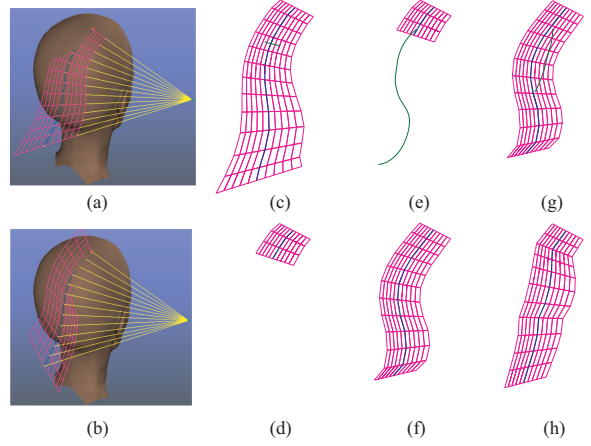


Figure 3: Editing a stream curve (blue). (a) and (b): supporting surface (purple) and the associated viewpoint of a stream curve. The supporting surface is either (a) a degenerate rule surface interpolating the stream curve and the associated viewpoint or (b) a strip composed of lines locally orthogonal to each triangle formed by a stream curve segment and the viewpoint. (c) and (d): cut operation: the hint stroke (green) is long and runs across the stream curve on the supporting surface; (e) and (f): concatenation operation: the hint stroke starts near the stream curve and ends far away from it, possibly out of the supporting surface (in which case the depth is extrapolated from the last segment on the supporting surface); (g) and (h): insertion operation: the hint stroke starts and ends near the stream curve.

and \mathbf{W}_- denote the matrices corresponding to $\bar{C} - C$ and $C - \bar{C}$, respectively. The new system matrix is

$$\bar{\mathbf{A}}^T \bar{\mathbf{A}} = \mathbf{D}^T \mathbf{D} + \mathbf{W}^T \mathbf{W} + \mathbf{W}_+^T \mathbf{W}_+ - \mathbf{W}_-^T \mathbf{W}_-.$$

The new Cholesky factor $\bar{\mathbf{L}}$ is computed by performing an update to \mathbf{L} :

$$\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T = \mathbf{L}\mathbf{L}^T + \mathbf{W}_+^T \mathbf{W}_+,$$

followed by a downdate to $\bar{\mathbf{L}}$:

$$\bar{\mathbf{L}}\bar{\mathbf{L}}^T = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T - \mathbf{W}_-^T \mathbf{W}_-.$$

Once the new Cholesky factor is computed, back-substitution is simply used to compute the vector field. We perform the incremental Cholesky factorization using an efficient sparse Cholesky factorization package [Dav07].

5. Sketch-based Hairstyle Design System

Our current implementation supports three kinds of style primitives: stream curve, dividing curve and ponytail. The ponytail primitive is a composite style consisting of four style curves. We represent a 3D style curve as a sequence of connected line segments. Each segment is of the same length as the discretization size of the vector field for convenience. These style curves are created and modified by sketching freeform strokes.

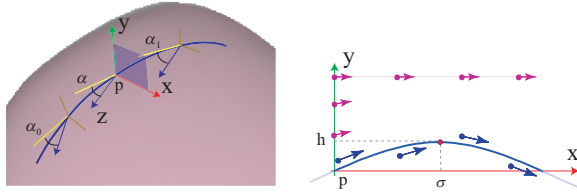


Figure 4: A dividing curve and its boundary constraints. **Left:** a local coordinate frame defined on the curve. Boundary constraints are defined in the local xy planes. The tilting angle α of a local plane is interpolated from two parameters α_0 and α_1 . **Right:** boundary constraints on a local xy plane. Blue vectors mimic the local shape of the parting hairstyle and follow the tangents of a Gaussian function, which passes through the origin \mathbf{p} at its inflexion point and with its local shape controlled by two parameters σ and h (set to 1 and 0.5 by default, with the unit as step size in the vector field). Red vectors explicitly represent the discontinuities around the parting line in the vector field and stop hair from growing across the parting line. Their magnitude is set smaller than 1 (we use 0.3) to reduce their global effect.

Depth determination is the main difficulty in 3D editing using 2D input devices. Our system relies on the scalp surface or a *supporting surface* associated with a style curve for depth determination.

Stream Curve. This is the simplest style primitive. It indicates the general flow direction of the hairstyle. Every segment of a stream curve will designate neighboring grid points as boundary constraints, each of which has its directional vector set as the direction of the segment's tangent (blue vectors in left of Figure 5). Optionally, to prevent hair growth beyond the end of the stream curve, we extrapolate and append a few empty segments. The magnitudes of the related directional vectors vanish gradually, from unit length to zero (red vectors in left of Figure 5).

A stream curve is created by drawing a stroke starting from the scalp. The depth of the starting point is the depth of the intersecting point on the scalp (Figure 3(a)). For each subsequent stroke point, if it is still on the scalp, it takes the depth of the scalp point, otherwise its depth is set as the depth of the last point on the scalp. A stream curve is usually long and needs incremental refinement from different viewpoints. To facilitate editing, its *supporting surface* is defined via expanding the stream curve in its neighborhood, and with respect to the viewpoint. We provide two ways of building the supporting surface, as illustrated in Figure 3(a)(b). User can freely change the viewpoint of a stream curve and/or switch between the two supporting surface modes to rebuild the supporting surface that is convenient for editing.

Once created, the user may edit a stream curve by sketching a *hint stroke* starting on its supporting surface. Depending on the general direction, starting and ending points, a

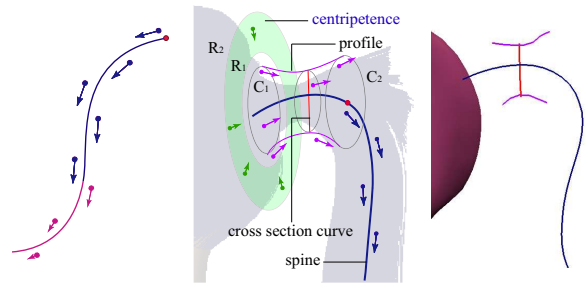


Figure 5: **Left:** a stream curve and its boundary constraints. **Middle:** a ponytail primitive and its boundary constraints. **Right:** four style curves of a ponytail primitive.

hint stroke is interpreted as three different editing operations (see Figure 3(c)-(h)).

Dividing Curve. Many hairstyles have a clear dividing line on the head where hair strands part and flow in opposite directions. A dividing curve is drawn on the scalp to represent such a line (see Figure 4). To model the discontinuities in the vector field around the parting line, a local Cartesian coordinate frame $\mathbf{L} = \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ is defined at every unit point \mathbf{p} on the dividing curve (Figure 4(a)), where \mathbf{y} is the normal of the scalp at \mathbf{p} , and \mathbf{z} is the tangent of the dividing curve at \mathbf{p} rotated by a tilt angle α that can be easily adjusted by the user. Boundary constraints are defined in the local xy planes, on two opposite sides of \mathbf{y} axis. The local shape of the boundary constraints can be easily adjusted via a few controlling parameters. See Figure 4 for details.

A dividing curve must be on the scalp and is relatively short, thus can usually be created by drawing one stroke on the scalp (inappropriate ones can be easily discarded). User can then draw a hint stroke on each side of the dividing curve (in the strip spanned by the yellow lines in Figure 2) to adjust the tilting angle of each local plane (see Figure 4).

Ponytail. The ponytail primitive consists of four style curves: a spine curve, a cross section curve, and two profile curves (Figure 5). The creation and editing of the spine curve is the same as that of a stream curve. The user then changes the viewpoint, rebuilds the supporting surface, and sketches the cross section curve and profile curves on it. A set of cross sections (circles) is then generated along part of the spine curve between the profile curves. Let C_1 and C_2 denote the first (near scalp) and the last cross sections respectively.

Three types of boundary constraints are derived (Figure 5). The first type (purple) are defined at grid points near to the cross sections, with their directions along the longitudinal directions of the revolutionary shape. The second type (blue) are at grid points near the spine curve. Their direction assignment is the same as for stream curves except that we start near the cross section C_2 (since fewer changes of boundary constraints means more efficient modification of Cholesky factorization). The third type (green), called cen-

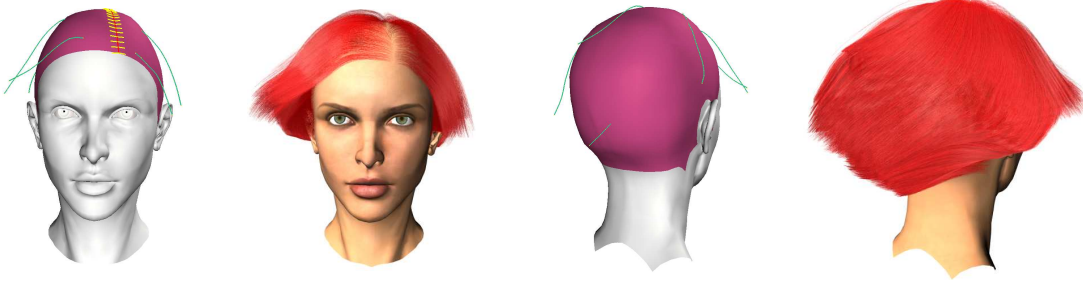


Figure 7: A hairstyle with a short parting line, created with seven stream curves and a dividing curve.

tripetal constraints, are introduced to make the vector field flow into the first cross section C_1 . They point to the center of C_1 and are defined at the region between two concentric circles R_1 and R_2 lying on the plane of C_1 . The radii of R_1 and R_2 are proportional to the radius of C_1 (we use 1.3 and 1.9). The magnitude of the last set of constraints is a parameter to control the tightness of the ponytail (we use the default value of 0.6). Essentially, the introduction of the centripetal constraints transfers the complexity of creating a ponytail from the user to the design of the ponytail sketch tool. Without the centripetal constraints, the user would need to draw many more stream curves to guide the hair curves to pass through the cross sections.

6. Implementation and Results

System Initialization The bounding volume and resolution of the discrete vector field is fixed during the whole design process. The vector field resolution depends on the desired hairstyle. A low resolution (about 25K variables) is used for smooth hairstyles, and high resolution (about 50K variables) for complex hairstyles such as a ponytail. The initialization of the linear system takes tens of seconds.

Hair Growth. To generate hair curves from the vector field, root points are first uniformly sampled on the scalp, with a small amount of randomness added. Two spherical coordinates $\theta \in [0, 180)$ and $\phi \in [0, 360)$ are used for scalp surface parameterization, and the hair density is controlled via the sampling steps of these two angles. Each hair curve is a sequence of connected line segments. In hair growth, let \mathbf{p} denote the current end point of a hair curve, a new segment

$l \cdot \mathbf{v}(\mathbf{p})$ is appended, where $\mathbf{v}(\mathbf{p})$ is the directional vector linearly interpolated from the vector field, and the scalar l takes a smaller value than the step size d in the vector field to make the growth smooth ($l = 0.5d$ in our implementation). Hair growth is terminated when $|\mathbf{v}(\mathbf{p})|$ is too small (< 0.05 in our implementation) or \mathbf{p} is out of the bounding volume.

Scalp Penetration Detection and Avoidance. It is not guaranteed that the flow lines in the vector field will not penetrate the scalp, which is undesirable and should be avoided. If necessary, the user can draw enough stream curves (usually about ten) around the scalp so that the flow lines in the vector field will not penetrate the scalp. We use a simple strategy to alleviate such non-essential user interaction. User first designs a hairstyle by drawing style curves without considering the scalp penetration problem. When a satisfactory hair shape is obtained, for each grid point near the scalp with direction pointing inwards[†], its direction is first replaced with its projection on the tangent plane of the nearest scalp point. All the grid points near the scalp are then added as boundary constraints. The magnitude of these directions are set small (we use 0.3) to reduce global influence. This step is done only once in the design process. This strategy is useful for saving user interactions when designing complex hairstyles such as ponytails.

Rendering. We use a free Renderman compliant software Aqsis [Aqs07] and the hair rendering algorithm proposed by Kajiya and Kay [KK89] to render all the results.

Results. Figure 6 shows a result created only using stream curves. Figure 7 demonstrates a typical hairstyle with a parting line. Figure 8 demonstrates a hairstyle with two ponytails. The hairstyle in Figure 1 uses eleven stream curves, four dividing curves and one ponytail primitive. All examples are created in a short time. The most complex hairstyle in Figure 1 takes five minutes.



Figure 6: A smooth hairstyle created with ten stream curves.

7. Conclusion

We present a hairstyle design system supported by an intuitive sketching interface and a fast vector field solver. User

[†] This can be easily detected by computing the angle between the direction and its nearest scalp normal.

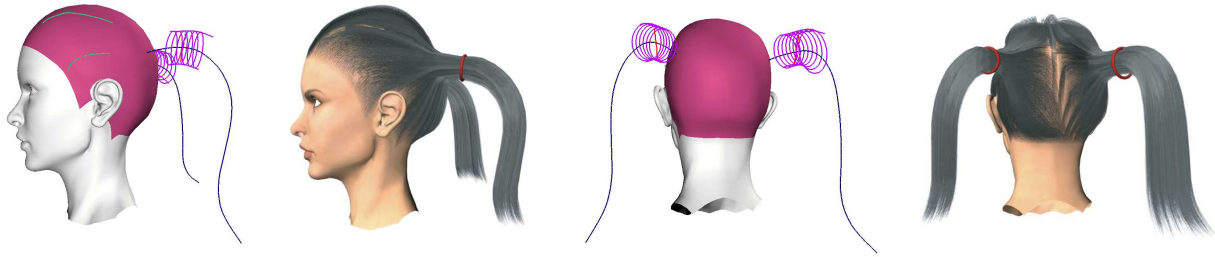


Figure 8: A hairstyle with two ponytails, created with four stream curves and two ponytail primitives. This is created in three minutes. See accompanying video for a demo.

can modify the hairstyle with interactive visual feedback. Interesting hairstyles can be easily designed.

Due to the smooth interpolation in the vector field, the output hairstyles of our system may lack local variations that are present in real human hair. There are two possible solutions. More style primitives may be designed to introduce high local variations, such as curls. Such an approach will require much higher resolution of the vector field and interactive response may become a problem. Another approach is to integrate previous hair modeling techniques as a post-processing step to add local details. Since the global shape and the positions of the hair curves are already satisfactory, post processing on such output is easier than designing hairstyles from scratch. Therefore, our system could be used as an independent design tool or a complementary pre-processor for other hair modeling techniques.

References

- [Aqs07] AQSIS., 2007. <http://www.aqsis.org/>.
- [BAC*06] BERTAILS F., AUDOLY B., CANI M.-P., QUERLEUX B., LEROY F., LÉVÊQUE J.-L.: Super-helices for predicting the dynamics of natural hair. *ACM Transaction on Graphics* 26, 3 (2006), 1180–1187.
- [CK05] CHOE B., KO H.-S.: A statistical wisp model and pseudophysical approaches for interactive hairstyle generation. *IEEE Transactions on Visualization and Computer Graphics* 11, 2 (2005), 160–170.
- [Dav07] DAVIS T.: Cholmod: a sparse supernodal cholesky factorization package., version 1.5, 2007. University of Florida, Available online at <http://www.cise.ufl.edu/research/sparse/cholmod/>.
- [DH99] DAVIS T. A., HAGER W. W.: Modifying a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications* 20, 3 (1999), 606–627.
- [GW97] GELDER A. V., WILHELMS J.: An interactive fur modeling technique. In *Proc. Graphics Interface* (1997), pp. 181–188.
- [HMT00] HADAP S., MAGNENAT-THALMANN N.: Interactive hair styler based on fluid flow. In *EG Workshop on Computer Animation and Simulation* (2000), pp. 87–99.
- [IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3D freeform design. In *SIGGRAPH '99* (1999), pp. 409–416.
- [KK89] KAJIYA J., KAY T.: Rendering fur with three dimensional textures. In *Proc. ACM SIGGRAPH '89* (1989), pp. 271–280.
- [KN02] KIM T.-Y., NEUMANN U.: Interactive multiresolution hair modeling and editing. *ACM Trans. Graph.* 21, 3 (2002), 620–629.
- [Mal05] MALIK S.: A sketching interface for modeling and editing hairstyles. In *EG Workshop on Sketch Based Interfaces and Modeling* (2005), pp. 185–194.
- [MKIA04] MAO X., KATO H., IMAMIYA A., ANJYO K.: Sketch interface based expressive hairstyle modelling and rendering. In *CGI '04* (2004), pp. 608–611.
- [MM06] MOON J. T., MARSCHNER S. R.: Simulating multiple scattering in hair using a photon mapping approach. *ACM Trans. Graph.* 25, 3 (2006), 1067–1074.
- [PoS04] PARIS S., O'H. M. B., SILLION F. X.: Capture of hair geometry from multiple images. *ACM Trans. Graph.* 23, 3 (2004), 712–719.
- [Sor06] SORKINE O.: Differential representations for mesh processing. *Computer Graphics Forum* 25, 4 (2006), 789–807.
- [WBC07] WITHER J., BERTAILS F., CANI M.-P.: Realistic hair from a sketch. In *Shape Modeling International* (June 2007).
- [WBK*07] WARD K., BERTAILS F., KIM T.-Y., MARSCHNER S., CANI M.-P., LIN M.: A survey on hair modeling: styling, simulation, and rendering. *IEEE Transaction on Visualization and Computer Graphics* 13, 2 (2007), 213–234.
- [WOQS05] WEI Y., OFEK E., QUAN L., SHUM H.-Y.: Modeling hair from multiple views. *ACM Trans. Graph.* 24, 3 (2005), 816–820.
- [Yu01] YU Y.: Modeling realistic virtual hairstyles. In *Pacific Graphics* (2001), pp. 295–304.
- [ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: SKETCH: An interface for sketching 3D scenes. In *Proceedings of ACM SIGGRAPH* (1996), pp. 163–170.