EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2005)
Takeo Igarashi, Joaquim A. Jorge (Editors)

A Sketching Interface for Modeling and Editing Hairstyles

Shahzad Malik

Department of Computer Science, University of Toronto, Toronto, ON, Canada
smalik@cs.toronto.edu

Abstract

This paper presents interaction techniques and algorithms for modeling and editing virtual 3D hairstyles with a user-friendly sketching interface. Using a pressure-sensitive tablet, a user makes freeform strokes to mimic a number of real-world hairstyling operations such as cutting, combing, curling, frizzing, and twisting. Additionally, the user can perform other localized operations such as implanting new hair strands onto a 3D surface (usually the scalp), lengthening the strands, and adjusting hair density. Virtual hairpins can also be placed onto strands to temporarily fix the position of hair which allows for the creation of more advanced styles such as ponytails. The system runs at interactive rates, thereby providing instant visual feedback to users as they work. Unlike existing hair modeling systems that require hours of complicated control point manipulations and parameter tweaking, our interface allows for creating expressive hairstyles quickly and easily, even for first-time users.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Modeling Packages; I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques.

1. Introduction

With computer-generated virtual characters now appearing in everything from blockbuster Hollywood films to 3D computer games, it is extremely important that these virtual humans each have their own unique identities and characteristics. It can be argued that hair is one of the most important physical attributes that we use to differentiate and characterize those around us, so it seems reasonable to assume that virtual characters should be created with equally realistic and varying hairstyles.

While researchers have made significant advancements in realistic virtual hair synthesis from a rendering and animation standpoint, modeling hair is still a difficult and time-consuming process. This is largely a consequence of the complex properties of hair, such as the large number of strands (typically 100,000 on an average human head), artificial styling techniques (gel, mousse, curlers, hairpins, etc.), and external forces (such as hair-hair collisions, hair-head collisions, gravity, and static charge).

In this paper we present a set of interaction techniques for modeling realistic and expressive hairstyles quickly and intuitively using a sketching interface. Using a tablet the user draws freeform strokes on and around a 3D head model to perform a number of hairstyling operations such as cutting, combing, curling, frizzing, and twisting. Due to the volumetric nature of hair, we leverage the pressure-sensing capabilities of tablets in order to quickly and fluidly access different layers of hair. In addition to the styling operations, hair can be implanted onto the 3D head, strands can be lengthened, and hair density can be modified, all using simple command strokes. Finally, sections of hair can be temporarily fixed in certain positions using virtual hairpins so that advanced styles such as ponytails can be created.

Figure 1 shows two hairstyles each created with our system in less than two minutes by an experienced user. Unlike most commercial 3D hair modeling tools that require time-consuming control point manipulations and parameter tweaking, our techniques are designed to allow exploration of a variety of hairstyles quickly and easily during the character design process. Our informal user feedback shows that even first-time users can create interesting hairstyles after only a few minutes of use.

Figure 1: *Hairstyles created in less than two minutes.*

2. Related Work

The graphics community has been interested in virtual hair modeling, animation, and rendering for the past two decades [MHK00]. Unfortunately most hair modeling systems described in the graphics literature make use of the ubiquitous Windows, Icons, Menus, and Pointers (WIMP) paradigm for their user interfaces. Thus for explicit hair modeling systems, where hair strands are defined individually, the user is required to manipulate control curves for each strand using the mouse in a graphical interface in order to generate a full head of hair [DM93]. Cluster/wisp hair systems [WS92, XY01, KN02] are somewhat more efficient from a user interface perspective since they require a user to only specify the control curve for an entire group of strands contained within a cylindrical region. In both cases, manipulating these control curves usually requires the user to edit control points in 3D space using a combination of viewpoints (perspective, side view, top view, front view), along with setting properties such as hair colour, density, or length using dialog boxes, pull-down menus, and scroll bars/sliders. As a result, many of these systems are extremely difficult to use without significant training, and even experienced artists typically require a few hours to design a detailed hairstyle. To make matters worse, the real-time editing windows in many of these applications usually only depict a coarse wireframe outline of the strands that is not representative of the final hairstyle. While a few do provide some higher-level editing operations such as cutting [LK01], combing [KN00], or curling, twisting, and braiding [KN02], they still rely on the user to perform these operations with a standard 3-button mouse which is cumbersome and unintuitive. Many of the popular commercial hair editing tools and plug-ins available today implement variations of the WIMP-based interfaces as described above.

Only recently have researchers considered developing more user-friendly interactive tools for modeling virtual hair. The HairPaint system described in [HR04] presents an alternative approach to modeling an explicit hair model by using a familiar 2D painting interface combined with a set of color scale images to specify hair characteristics. Therefore by drawing in some particular color on a flattened 2D representation of a 3D head model, geometrical attributes such as hair root position, density, and hair length can be established. While the system allows interesting hairstyles to be created fairly quickly, there are still some shortcomings. From a user interface perspective, having to draw in a 2D canvas window while observing the corresponding changes in a 3D view can be confusing as well as inefficient, since it requires the user to continuously change the focus of attention. Additionally, drawing on the flattened 2D representation of the head does not allow for accurate hair placement or hair orientation since the mapping between the 2D and 3D views is indirect. This results in the user having to frequently make guesses as to where a point on the 2D canvas maps to the 3D view.

Instead of manipulating individual strands or clusters, the vector-field and fluid-flow hair systems presented in [Yu01] and [HM00] propose a radically different approach to hair modeling. A user interactively places a number of field primitives such as streams, sources, and vortexes around a 3D head. These primitives are then used to control the growth direction of hair emanating from the scalp in a post-process. While the final rendered results are impressive, these field-based approaches make it almost impossible to generate artificial effects such as braids or ponytails. Additionally, the hair generation process takes a significant amount of processing time, so it is difficult to determine how the final hairstyle will look just from the positioning of the primitives.

Recent work has shown that it is possible to capture hair from multiple images of a real hairstyle using computer vision techniques [GSML02, PBS04, WOQS05] which opens up the possibility of completely automating the hair design process. While the current results are impressive, the systems require about an hour of processing time to reconstruct the hairstyles. Additionally the existing systems have difficulty capturing complex styles such as spiky hair, braids, or ponytails. Therefore efficient interactive tools are still required to complement these computer vision techniques in order to enhance or modify the captured styles.

Our hair sketching interface is motivated by recent work that shows intuitive techniques for the rapid exploration and design of 3D models [IMT99, TBSR04], clothing manipulation [IH02], garment design [TCH04], and character animation [Osh04]. Mao et al. [MKKI02, MKIA04] were the first to show freeform strokes being used to model hair, but their system is extremely limited in the hairstyles it is capable of generating since all hairstyles are assumed to be symmetric about a partition line located at the center of the top of the scalp. In other words, the user draws a partition line on the scalp, followed by a single silhouette line on one side of the head. The system then generates a symmetric hairstyle based on these input strokes. As a result, short hairstyles such as a Mohawk are impossible to create with their system. Also, fine editing of the generated hairstyles is not possible since hair is generated in a single shot. In contrast, our system allows

for hair to flow in any user defined direction and without any restrictions on how the hair strands are distributed on the scalp. Additionally, our system allows for advanced editing operations to be performed at any time such as lengthening, cutting, combing, curling, frizzing, and twisting.

3. Hair Representation

In our system, hair is represented using clusters as first proposed in [XY01]. Therefore large groups of strands can be manipulated simultaneously by simply adjusting the control curve for the cluster that contains them (Figure 2). We use generalized cylinders to represent the boundary of hair clusters, while 3D Catmull-Rom splines are used to represent the underlying control curve since they guarantee that the curve will pass through the corresponding control points. The control curve passes through the center of the generalized cylinder, with the root of the control curve located on some surface (usually the scalp of a 3D head model). We assume the polyline consisting of the curve control points has equal length segments of 2cm, which provides a nice tradeoff between curve smoothness and efficient processing time for interactive control.

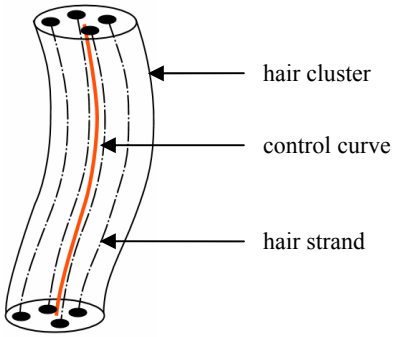

Figure 2: *A group of strands is contained within a hair cluster whose shape is controlled by a curve.*

Each cluster has a set of parameters that affect the distribution of the strands contained within it. The first parameter is *density*, which ranges from 0 to 1. A density value of 0.0 denotes no hair strands inside of the cluster, while a value of 1.0 denotes a maximal number of hair strands (Figure 3). We currently set the maximum number of hair strands per cluster to 512, which works reasonably well for our fixed-size clusters with a radius of 1cm.

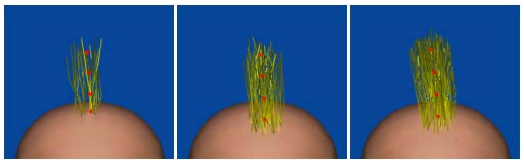

Figure 3: *Left to right: low density cluster (0.1), medium density cluster (0.5), high density cluster (1.0).*

The second cluster parameter is *twist*, which is a rotation value (in degrees) set per control point. This twist parameter allows cross-sections of the cluster to be rotated about the corresponding control point as shown in Figure 4 and Figure 5.

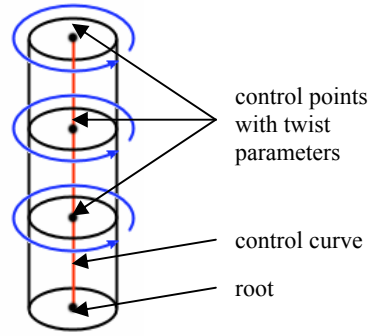

Figure 4: *Twisting is specified at each curve control point (except the root), allowing strands to be locally rotated about the control curve.*

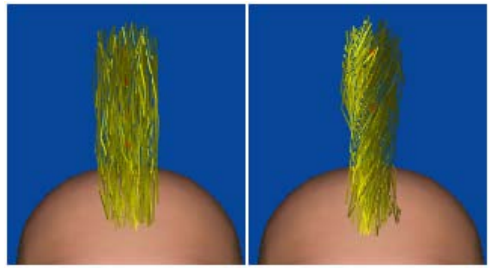

Figure 5: *(Left) Cluster with twisting set to 0; (Right) Cluster with twisting set to 45 degrees for every control point.*

Hair clusters also have a *frizziness* parameter ranging from 0 to 1 that is used to control the amount of shape variation of strands within the cluster. A frizziness of 0.0 denotes no shape variance, while a value of 1.0 denotes high shape variance (Figure 6).

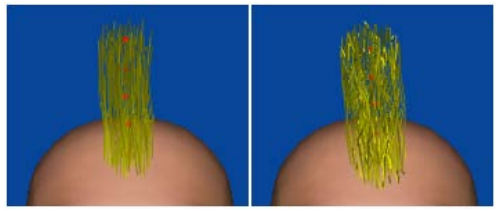

Figure 6: *(Left) Cluster with low frizziness (0.1); (Right) Cluster with high frizziness (0.9).*

Based on these parameters, hair strands can be generated inside of each hair cluster using the following equation:

$$p_k = \text{twist}(c_k + d_k, \sum_{i=1}^{k} \theta_i)$$

where $\mathbf{p_k}$ is the k-th control point for some hair strand, $\mathbf{c_k}$ is the k-th control point for the control curve, $\mathbf{d_k}$ is a displacement from $\mathbf{c_k}$, θ_k is the k-th twist value in degrees, and **twist** is a function to rotate $\mathbf{c_k}+\mathbf{d_k}$ about the axis formed from $\mathbf{c_{k-1}}$ to $\mathbf{c_k}$.

The displacement $\mathbf{d_k}$ is computed iteratively similar to the approach described by Choe and Ko [CK05]:

$$\mathrm{d}_k = \mathrm{d}_{k-1} + \mathrm{e}$$

where **e** is a noise vector. To ensure that $|\mathbf{d_k}|$ is less than the radius r of the cluster, we compute $\mathbf{e}=y\mathbf{x}$, where **x** is a random unit vector and y is randomly chosen from the range $[0, \min(L,\sigma \cdot r)]$ where L is the intersection of the vector **e** with the cluster's generalized cylinder, and σ is the cluster's frizziness value.

Note that we still need to compute the initial displacement of hair strands $\mathbf{d_0}$. Assuming the root position $\mathbf{c_0}$ of the control curve is located on the surface of a 3D head model, we uniformly distribute the desired number of strand root positions $\mathbf{p_0}$ (based on the density parameter) around $\mathbf{c_0}$ so that they remain within the cluster's radius. To guarantee that these new strand roots emanate from the scalp as well, we use a pre-computed distance field [JS01]. A distance field provides a fast lookup of the distance and gradient to the closest point on a 3D model for some query point. Therefore for each new strand root position $\mathbf{p_0}$, we determine the distance d to the closest point on the 3D model, and if the distance is not zero we extract the gradient ∇ to the closest point and set the strand's new root position to:

$$\mathrm{p}_{new} = \mathrm{p}_0 - d\nabla$$

The distance field is also used in a similar manner to prevent cluster and strand control points from penetrating the 3D head model.

4. Hair Sketching User Interface

In this section we describe the hair sketching interface from a user's perspective, along with the corresponding algorithms for creating and styling hair clusters. We assume the system is being used with a pressure-sensitive tablet along with a stylus that has at least one button along its edge and a pressure-sensitive eraser tip. The interface features a single perspective view of a 3D head model that can be rotated using the virtual trackball technique described in [Hul90] by pressing the button on the edge of the stylus. The tablet input space is mapped such that the corners of the interface window are mapped to the corners of the tablet. Hair is rendered on the 3D head model in real-time with realistic lighting and shadows to provide instant feedback on the results of styling operations. The shape of the cursor in the interface window dynamically changes based on the action the user is performing (see Figure 7). Figure 8 shows a snapshot of the user interface, which consists of the 3D head model, a cursor, and three buttons (load, save, and exit).

Figure 7: *The cursor changes dynamically based on stylus position and/or pressure. From left to right: default cursor, implanting cursor, cutting/density cursor, styling cursor, and color selection.*

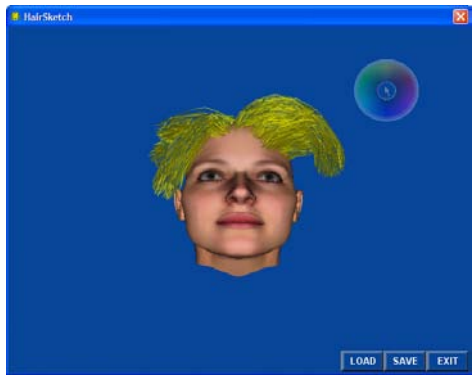

Figure 8: *A snapshot of the hair sketching interface. The cursor is currently in hair color selection mode.*

Hairstyling operations are performed using the tip of the stylus with varying amounts of pressure. Ramos et al. [RBB04] showed that dividing the pressure range of a tablet into six or less discrete levels instead of using it as a continuous value produces the best user performance and control. We leverage this information by defining three equal pressure levels for performing styling operations in different layers of hair:

Level 1 (light pressure): Styling operations occur on the hair cluster closest to the stylus tip's cursor position in the perspective view. The closest cluster is determined by casting a ray through the cursor position in the viewing plane, and computing the closest intersection point with a hair cluster.

Level 2 (medium pressure): Styling operations occur on all hair clusters that are intersected by the ray cast through the stylus tip's cursor position. The intersection point with the 3D model is also computed, and only clusters between the viewer and the 3D model are considered.

Level 3 (heavy pressure): Styling operations occur on the surface of the 3D model.

In all cases, the pressure level and affected clusters are determined based on the average pressure at stroke locations within 2 pixels of the stroke's start position. Therefore as a stroke is drawn, subsequent changes in

pressure do not alter the pressure level that was detected at the start of the stroke.

In the following sections we describe how the user can interactively implant hair onto the head, lengthen and cut strands, adjust hair density, and style/comb hair in various ways. The user is provided with continuous visual feedback based on the position of the cursor and the pressure of the stylus on the tablet.

4.1 Hair Implanting

To place hair onto the 3D head model, the user moves the cursor tip to the desired location on the scalp from which strands should emanate. With the pressure at level 3, the user then draws a freeform stroke that defines how the hair should flow. When the tip of the stylus is raised above the tablet surface, the root position **r** of the cluster is determined by computing the intersection point between the 3D head model and a ray going through the stroke's start position on the image plane. The system then generates a hair cluster that follows the user defined curve as shown in Figure 8.

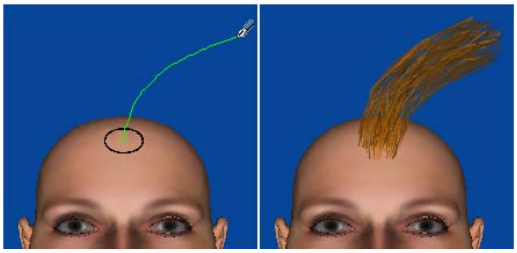

Figure 8: *Implanting a hair cluster.*

In the case where the selected scalp location lies within an existing hair cluster, the existing cluster is deleted and a new cluster is created in its place based on the input stroke.

Visual feedback is provided to the user based upon the pressure level. At pressure level 3, all existing hair on the 3D model is rendered semi-transparently so that the scalp is visible. All new clusters are created with a fixed radius, and the user can see an approximate outline of the cluster's position and dimensions on the scalp surface before beginning the stroke.

Converting a 2D stroke into a 3D curve for the hair cluster is an illposed problem, since there is an entire family of 3D curves with different depth variations that can all map to the same 2D stroke. To remedy this ambiguity somewhat, we assume that hair exhibits some spatial coherence. Therefore, if there are other hair clusters closeby that have control curves which flow in similar directions to the user-drawn 2D stroke, then the system averages the 3D orientation of those nearby clusters to determine the 3D representation of the new hair cluster. This allows for faster sketching of new clusters, since an initial cluster can be positioned as desired using detailed styling strokes (as described in Section 4.5), while subsequent clusters (which users typically draw close to previously placed clusters) can be created with single implanting strokes without the need for detailed styling. If there are no nearby clusters with similar 2D control curves, the new cluster is created assuming the 3D curve lies in the view plane, with the depth value based on the position of the cluster's root on the scalp. Figure 9 shows the benefit of this feature.

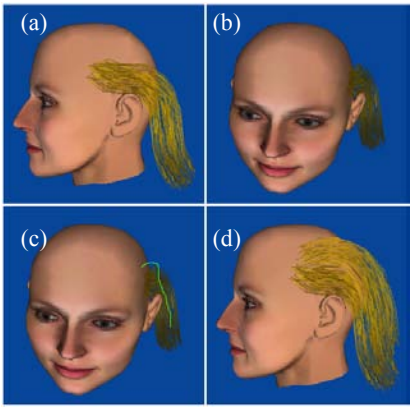

Figure 9: *3D curve estimation based on neighboring hair clusters: (a) An initial cluster viewed from the side; (b) A front view of the initial cluster; (c) A 2D stroke is drawn from the front view that flows similarly to the initial cluster; (d) The new 3D cluster flows similarly to the initial cluster.*

Before converting the 2D input stroke into a 3D Catmull-Rom spline, the input stroke is first converted into a 2D polyline with equal length segments. All other existing hair clusters are then transformed into camera space and projected into image space. The new cluster's first spline control point $\mathbf{c_0}$ is then set to **r**, while subsequent control points $\mathbf{c_i}$ are computed iteratively as follows:

- Extract the endpoints of the 2D polyline segment p_{i-1} and p_i.
- Compute the normalized orientation vector $\hat{p}_i$ for the polyline segment $\mathrm{p}_{i-1}\mathrm{p}_i$.
- From each cluster *j*, find the closest control point d_a^j to c_{i-1} in camera space below some maximum distance $\boldsymbol{\tau}$, if one exists.
- Compute the 2D image points q_{a-1}^j and q_a^j by projecting d_{a-1}^j and d_a^j.
- Compute the normalized orientation vector $\hat{q}_j$ for the 2D vector $\mathrm{q}_{a-1}^j\mathrm{q}_a^j$, discarding those vectors with zero length.

- If the angle θ_i between $\hat{p}_i$ and $\hat{q}_j$ is above some threshold (we use 30 degrees), we discard the corresponding d^j_{a-1} and d^j_a control points.
- Using all other *N* remaining closest control points, we compute

$$c_i(z) = c_{i-1}(z) + b_i \|p_i - p_{i-1}\|$$

where

$$b_i = \frac{\sum_j^N \left[\left(1.0 - \frac{\|c_{i-1} - d^j_a\|}{\tau} \right) \cos(\theta_i) \left(\frac{d^j_a(z) - d^j_{a-1}(z)}{\|d^j_a - d^j_{a-1}\|} \right) \right]}{\sum_j^N \left(1.0 - \frac{\|c_{i-1} - d^j_a\|}{\tau} \right)}$$

- In the case where there are no valid nearby control points from other clusters, set $c_i(z) = c_{i-1}(z)$.
- Back project the image point at p_i using $c_i(z)$ as the depth value to retrieve the 3D coordinates for $\mathbf{c_i}$.

The basic idea behind this algorithm is to compute a weighted average of the depth variations of neighboring clusters that flow in a similar direction to the input stroke (in 2D). This weighted average can then be used to compute the depth variations for the new cluster. Note that our formulation interpolates depth values linearly in image space, which is inaccurate. However, for our chosen control curve segment lengths we have found this approach to work sufficiently well in practice.

Once the control points for a new cluster have been determined, a simple smoothing operation is performed to eliminate sharp angles in the corresponding 3D polyline. Individual hair strands are then generated from the control curve as described in Section 3.

4.2 Lengthening Hair

Existing hair clusters can be lengthened by placing the cursor near the end of the desired cluster and drawing a stroke with pressure level 1. For visual feedback, the closest cluster becomes highlighted with a green tint as soon as pressure is applied to the tablet, to help the user determine the end position of the cluster. When the stylus tip is raised above the tablet surface, the system grows the cluster so that it follows the path of the drawn stroke (Figure 10). As with implanting, the lengthened 3D segments of the cluster are determined by averaging any existing nearby clusters that have similar 2D control curves.

Figure 10: Lengthening a hair cluster.

4.3 Cutting Hair

Hair can be cut by drawing a relatively straight stroke across the boundary of a hair cluster using the pressure-sensitive eraser tip. The stroke must start on one side of the cluster and then exit on the other side in image space in order for the cutting operation to be performed. The cutting operation effectively works by computing the intersection between the drawn stroke and a cluster's control curve in image space (Figure 11). At pressure level 1, only the closest hair cluster that is intersected by the stroke in the perspective view is cut. If the stroke is drawn at pressure level 2, all clusters that are intersected by the stroke down to the scalp are cut. A cutting stroke at pressure level 3 completely removes any hair clusters intersected on the scalp surface. Before a cluster is cut, its existing hair strands are removed. After the cut, a new set of hair strands is computed based on the existing cluster parameters, as described in Section 3.

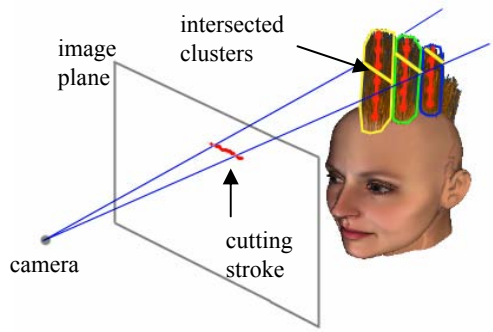

Figure 11: *Cutting hair clusters.*

4.4 Controlling Hair Density

By default, all new hair clusters are created so that their density is a weighted average of the densities of neighboring hair clusters. The weight assigned to each neighbor is based on the distance between the root positions so that closer clusters are weighted higher. In the case where there are no proximal clusters, density is set to a default value of 0.5. To interactively increase the density of hair on existing clusters, the user draws a clockwise circular stroke with the eraser tip. At pressure level 1, the hair cluster closest to the center of the drawn circle in the perspective view has its density value increased by a fixed amount (we currently use an increment of 0.05). A counter-clockwise circle decreases the density value. At pressure level 2, all clusters

contained within the drawn circular stroke have their densities increased or decreased. Finally, at pressure level 3 only the clusters emanating from the scalp area that falls inside of the drawn circle have their densities increased or decreased. Similar to the way cutting works, hair density adjustments are accounted for by first removing all hair strands from the cluster and then recomputing new strands as described in Section 3 (which takes the new density parameter into account). Unlike previous operations, however, density adjustments are performed each time a full circle is drawn, regardless of whether or not the stylus has been raised off of the tablet surface (Figure 12).

Figure 12: *Adjusting hair density. A full circle establishes density adjustment mode, while subsequent circles in the same stroke continuously modify density.*

4.5 Styling Hair

Our system allows a user to perform a number of advanced styling operations on an existing set of clusters to refine the look of a hairstyle. The first such styling operation is *twisting*, which causes strands in a cluster to locally rotate about the cluster's control curve as described in Section 3. The twisting parameters for a cluster can be interactively modified by drawing a twirl stroke with the tip of the stylus as shown in Figure 13. When the stroke is completed, the system first determines the clusters to affect based on the stroke's start position and the pressure level. The system then determines the points where the stroke overlaps itself. For each of these overlap positions, the distance to the closest control point in each affected cluster is found. If an overlap position falls within the silhouette of an affected cluster, the closest control point's twist parameter is adjusted by 15 degrees. A clockwise twirl stroke increments the twist parameter, while a counterclockwise twirl stroke decrements the twist. Therefore by adjusting the frequency of twirls the strength of the twisting can be interactively controlled.

Frizzing is a hairstyling technique used to introduce small, tight curls onto individual strands. In our system the user can control the frizziness parameter of a hair cluster by drawing a scribble stroke as shown in Figure 14. The affected clusters are determined based on the pressure level and position at the start of the scribble stroke. If the average amplitude for the scribble is greater than or equal to the projected diameter of the affected cluster, the frizziness is set to 1. If the average amplitude is less than or equal to one-quarter of the projected diameter, the frizziness is set to 0. All average amplitudes in between these extremums linearly map to intermediate frizziness values. Unlike twisting, frizzing adjustments affect a cluster globally as outlined in Section 3.

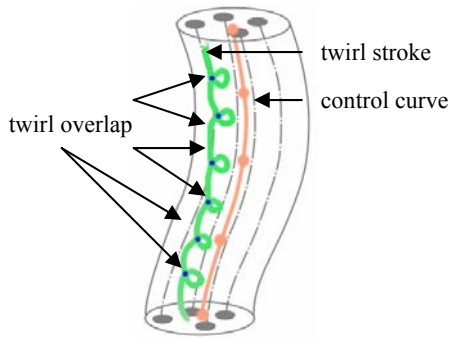

Figure 13: *Twisting hair. Each overlap of the twirl stroke contributes a 15 degree rotation to the closest control point for the cluster.*

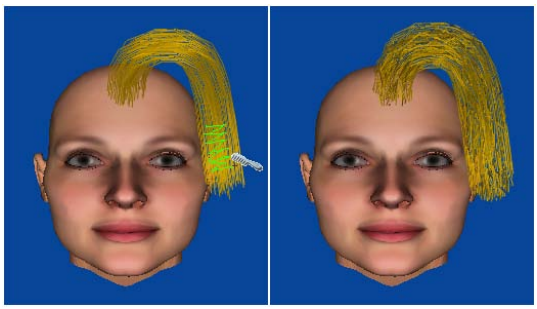

Figure 14: *Frizzing hair.*

Hair can also be combed or curled by drawing smooth strokes that act as forces which push or deform the control curve of the underlying hair clusters in the viewing plane. Since such combing strokes are similar to those used to implant hair, pressure is used to determine what action the user wishes to perform. At pressure level 1, combing only affects the cluster closest to the start of the stroke in the viewing plane, while at pressure level 2 all clusters that are intersected by the ray through the start of the stroke are affected. Pressure level 3 is used to implant hair on the scalp as described earlier.

To deform the control curve of affected clusters, the system first converts the input stroke into a polyline with uniform length segments of size ρ (we currently use ρ=30 pixels which works quite well). Each segment of this polyline then acts as a force vector that slightly nudges cluster control points located within ρ pixels from the segment start position. Let $\mathbf{v_i}$ represent the normalized direction vector for segment i of the input stroke, and $\mathbf{q_i}$ the start position of segment i. Let $\mathbf{c_j}$ represent the j-th control point of an affected cluster (not the root), and let $\mathbf{p_j}$ represent the projection of $\mathbf{c_j}$ on the image plane. We first compute the distance d between $\mathbf{q_i}$ and $\mathbf{p_j}$. If $d > \rho$ then we consider $\mathbf{p_j}$ to be out of the influence range of $\mathbf{v_i}$. If it is less than ρ, we compute w=1.0-d/ρ as the attenuated force magnitude. $\mathbf{p_j}$ is then displaced by $w\mathbf{v_i}$ as shown in Figure 15, and then back projected to compute a new $\mathbf{c_j}$ (using the depth value of the original $\mathbf{c_j}$). After all control

points for a particular cluster have been displaced by the appropriate force vectors, the system performs relaxation steps on the control curve's polyline so that segment lengths are approximately uniform and there are no sharp angles. Adjustments are also made to prevent penetrations with the 3D model as discussed in Section 3. Finally, hair strands are regenerated for the entire cluster based on the cluster parameters.

While the current approach of using fixed-size segments for our combing forces works well in most instances, there is a tradeoff to be made with respect to speed and accuracy. Clearly, we could use much larger segments to increase speed, but this increases the chance that a combing stroke will have little or no effect on a cluster. Similarly, decreasing the segment lengths would improve the effect of the combing operation, but processing time would dramatically increase, preventing the system from operating in real-time.

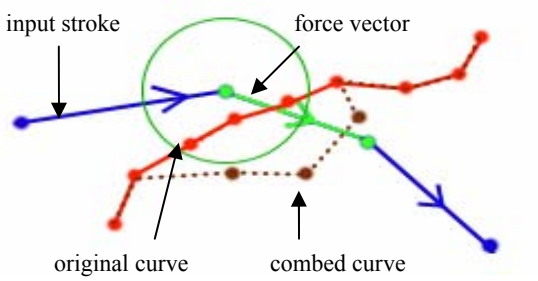

Figure 15: *Combing hair.*

4.6 Virtual Hairpins

The combing operation can potentially affect the entire length of a hair cluster depending on the size of the user-drawn stroke. Therefore only the root of the cluster is guaranteed to remain in its original location. In certain instances the user may desire other sections of a hair cluster to remain in-place. By double tapping the tip of the stylus on some part of a cluster, a virtual hairpin is activated which locks the position of the closest control point for the underlying cluster much like the root. Subsequent combing operations will not affect the cluster between two hairpin locations, allowing for the creation of styles such as ponytails which will maintain their shape as long as the hairpins are active. Hairpins are rendered on-screen as simple green highlights (Figure 16). A hairpin can be removed by simply double tapping it again with the tip of the stylus.

The system maintains hairpin constraints during the combing relaxation phase. At each iteration, the control points that lie between two hairpinned locations are simply moved back to their locked positions so that the remaining segments of the control curve can converge to a smooth state.

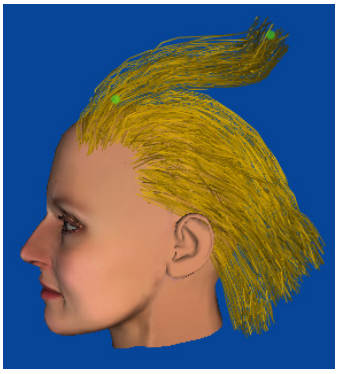

Figure 16: *Hairpins (denoted as green highlights) can be used to lock the positions of clusters.*

4.7 Changing Hair Color

If the cursor is moved to a position on the screen at which there are no 3D objects (hair or head) within a 30 pixel radius, a semi-transparent color wheel is drawn around the cursor (Figure 17). This color wheel can be used to select the hair color for subsequent implanting operations. The color wheel moves with the cursor as long as pressure remains in the level 1 range and the cursor is still sufficiently far away from any 3D geometry. If pressure increases into the level 2 or 3 range, the color wheel appears fully opaque and detaches from the cursor so that it no longer moves with it. The user can therefore select a color by first applying and holding medium to high pressure with the stylus, selecting a color on the color wheel, and then releasing pressure. While a fixed color wheel in some region of the screen could provide similar functionality, we feel that attaching it to the cursor leads to a less cluttered interface as well as reduces the number of focus changes required by the user.

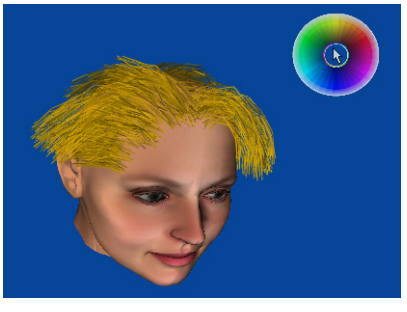

Figure 17: *Color wheel for changing the active color.*

5. Implementation and Results

The system was implemented in C++ under Windows XP, using OpenGL for rendering and a Wacom Graphire3 pressure-sensitive tablet for input. We used the shading model described by Kajiya and Kay [KK89], along with opacity maps for hair shadowing [KN01]. Hair strands were rendered as anti-aliased OpenGL lines in a back-to-

front order similar to the approach described in [KN02]. Interactive frame rates were achieved throughout the modeling and editing process using a PC equipped with a Pentium 4 processor running at 3.0 GHz and an ATI Radeon 9800 video card.

While we have not yet performed any detailed usability study of our interface, we did allow three professional 3D artists to use the system in order to gauge their feedback. Each of the artists had experience in character design using a number of popular commercial modeling tools, including basic experience with some existing hair design plugins. After a brief 5 minute introduction to our hair sketching interface, each artist was allowed to freely use the system. Within 5-10 minutes each artist was creating interesting and detailed styles with relative ease, using about 20-30 clusters. Figure 18 shows three actual hairstyles created by each of the artists. The artists all commented that the interface was much more intuitive and required significantly less time to create detailed styles than the hair tools they had used in the past.

While overall feedback was positive, one of the artists felt that the cutting operation wasn't as precise as he would like it to be when making cutting strokes that were non-orthogonal to the cluster. This is a result of the cutting stroke only affecting the underlying control curve of a hair cluster instead of the actual strands. Regarding the combing operation, one of the artists commented that in some instances it would be nice if it also took neighboring cluster directions into account much like the implanting and lengthening operations, since the current implementation (which combs in the viewing plane) occassionally leads to strange styles when observed from different viewpoints. Another minor gripe shared by all artists was that it was sometimes difficult to implant all of the initial clusters when creating a new hairstyle from scratch. They felt that a simple tool to quickly copy an existing cluster to other parts of the head would further reduce the time needed to create detailed hairstyles.

Figure 18: *Three different hairstyles created by first-time users in under 10 minutes.*

6. Conclusions and Future Work

In this paper we presented interaction techniques and algorithms for quickly modeling detailed hairstyles using an intuitive sketching interface. While our tool achieves its goal of allowing artists to quickly explore a number of different hairstyles when designing 3D characters, there are still areas where the system can be improved. In particular, the use of hair clusters prevents the user from controlling individual hair strands, which is useful for adding fine details. The hierarchical cluster model presented by Kim and Neumann [KN02] would help resolve this limitation, but it is not immediately clear what sort of interactions or gestures could be used to easily traverse the different detail levels.

Hair-hair collision is currently ignored in our system since the extra processing power it requires would prevent the system from operating in real-time. Nevertheless, it would be beneficial to incorporate an efficient hair-hair collision algorithm into the current system to further increase the realism of the generated hairstyles.

Our cluster generation algorithm currently assumes that the radius remains fixed for the entire generalized cylinder. This reduces the realism of locks or ponytails since hair strands do not come together at the tips as expected. While the cluster representation presented in [CK05] can be used to facilitate such styles, we must still develop the interaction techniques that can provide an intuitive interface to these extra control parameters. One possible solution is to allow clusters to be created by drawing their silhouettes instead of just specifying the internal skeleton curve. This would be similar to the extrusion operation presented in [IMT99] where the generalized cylinder of a hair cluster could be swept along the outline of a curve whose endpoints both emanate from locations on the scalp.

While our user interface currently only allows a user to rotate the viewpoint around the 3D model, it is reasonably simple to incorporate translation and zooming to the viewpoint control. This would be useful for creating large furry creatures or animals, since our system is general enough to allow hair to be placed on any arbitrary 3D model.

7. Acknowledgements

We would like to thank Allan Jepson, Joe Laszlo, and Abhishek Ranjan for thoughtful discussions. Financial support by the Natural Science and Engineering Research Council of Canada (NSERC) and the Government of Ontario is gratefully acknowledged.

References

[CK05] CHOE B., KO H-S.: A Statistical Wisp Model and Pseudophysical Approaches for Interactive Hairstyle Generation. In *IEEE Transactions on Visualization and Computer Graphics*, 11(2), 2005, pp. 160-170.

[DM93] DALDEGAN A., MAGNENAT-THALMANN N.: Creating Virtual Fur and Hair Styles for Synthetic Actors". In *Communicating with Virtual Worlds*, Springer-Verlag, 1993, pp. 358-370.

[GSML02] GRABLI S., SILLION F., MARSCHNER S., LENGYEL J.: Image-based Hair Capture by Inverse Lighting. In *Proceedings of Graphics Interface*, 2002, pp. 51-58.

[HM00] HADAP S., MAGNENAT-THALMANN N.: Interactive Hair Styler based on Fluid Flow. In *Proceedings of Eurographics Workshop on Computer Animation*, 2000, pp. 87-100.

[HR04] HERNANDEZ B., RUDOMIN I.: Hair Paint. In *Proceedings of IEEE Computer Graphics International* (CGI), 2004. pp. 578-581.

[Hul90] HULTQUIST J.: A Virtual Trackball. *Graphics Gems* (ed. A. Glassner), Academic Press, 1990, pp. 462-463.

[IH02] IGARASHI T., HUGHES, J. F.: Clothing Manipulation. In *Proceedings of ACM UIST*, 2002, pp. 91-100.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A Sketching Interface for 3D Freeform Design. In *Proceedings of ACM SIGGRAPH*, 1999, pp. 409-416.

[JS01] JONES M., SATHERLEY R.: Using Distance Fields for Object Representation and Rendering. In *Proceedings of Eurographics Annual Conference* (UK Chapter), 2001, pp. 37-44.

[KK89] KAJIYA J., KAY T.: Rendering fur with three-dimensional textures. In *Proceedings of ACM SIGGRAPH*, 1989, pp. 271-280.

[KN00] KIM T-Y., NEUMANN U.: A Thin-Shell Volume for Modeling Human Hair. In *Proceedings of IEEE Computer Animation*, 2000, pp. 104-111.

[KN01] KIM T-Y., NEUMANN U.: Opacity Shadow Maps. In *Proceedings of Eurographics Workshop on Rendering Techniques*, 2001, pp. 177-182.

[KN02] KIM T-Y., NEUMANN U.: Interactive Multiresolution Hair Modeling and Editing". In *Proceedings of ACM SIGGRAPH*, 2002, pp. 620-629.

[LK01] LEE D-W., KO H.: Natural Hairstyle Modeling and Animation. In *Graphical Models*, 2001, 63(2), pp. 67-85.

[MHK00] MAGNENAT-THALMANN N., HADAP S., KALRA P.: State of the Art in Hair Simulation. In *Proceedings of International Workshop on Human Modeling and Animation*, 2000, Korea Computer Graphics Society, pp. 3-9.

[MKKI02] MAO X., KASHIO K., KATO H., IMAMIYA A.: Interactive Hairstyling Modeling using a Sketching Interface. In *Proceedings of International Conference on Computer Science* (ICCS), 2002. Lecture Notes in Computer Science (LNCS) 2330, pp. 131-140.

[MKIA04] MAO X., KATO H., IMAMIYA A., ANJYO K.: Sketch Interface Expressive Hairstyle Modeling and Rendering. In *Proceedings of IEEE Computer Graphics International* (CGI), 2004, pp. 608-611.

[Osh04] OSHITA M.: Pen-to-mime: A Pen-based Interface for Interactive Control of a Human Figure. In *Proceedings of Eurographics Workshop on Sketch Based Modeling*, 2004, pp. 43-52.

[PBS04] PARIS S., BRICENO H., SILLION F.: Capture of Hair Geometry from Multiple Images. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 2004, pp. 712-719.

[RBB04] RAMOS G., BOULOS M., BALAKRISHNAN R.: Pressure Widgets. In *Proceedings of ACM CHI*, 2004, pp. 487-494.

[TBSR04] TSANG S., BALAKRISHNAN R., SINGH K., RANJAN A.: A Suggestive Interface for Image Guided 3D Sketching. In *Proceedings of ACM CHI*, 2004, pp. 591-598.

[TCH04] TURQUIN E., CANI M-P., HUGHES J. F.: Sketching Garments for Virtual Characters. In *Proceedings of Eurographics Workshop on Sketch Based Modeling*, 2004. pp. xx-xx.

[WS92] WATANABE Y., SUENAGA Y.: A Trigonal Prism-based Method for Hair Image Generation. In *IEEE Computer Graphics and Applications*, Volume 12, Issue 1, 1992, pp. 47-53.

[WOQS05] WEI Y., OFEK E., QUAN L., SHUM H-Y.: Modeling Hair from Multiple Views. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 2005. *To appear*.

[XY01] XU Z., YANG X-D.: V-HairStudio: An Interactive Tool for Hair Design. In *IEEE Computer Graphics and Applications*. Volume 21, Issue 3, 2001, pp. 36-43.

[Yu01] YU Y.: Modeling Realistic Virtual Hairstyles. In *Proceedings of Pacific Conference on Computer Graphics and Applications* (PG), 2001, pp. 295-304.