Example-Based Hair Geometry Synthesis

Lvdi Wang[1] [1]Tsinghua University
Yizhou Yu[2] [2]University of Illinois at Urbana-Champaign
Kun Zhou[3] [3]Zhejiang University
Baining Guo[1,4] [4]Microsoft Research Asia

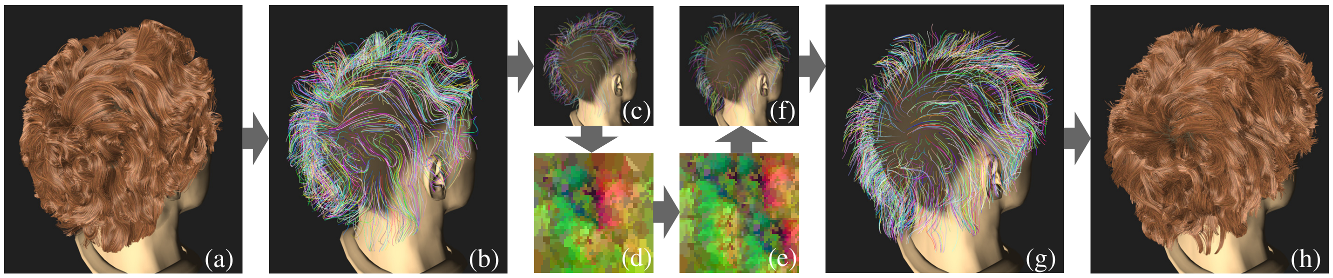


Figure 1: *Example-based hair geometry synthesis pipeline. (a) input hair geometry (level-0); (b) level-1 and (c) level-2 geometry of the input hierarchy built upon (a); (d) 2D feature map generated from (c); (e) output feature map generated from (d) using 2D texture synthesis; (f) level-2 geometry reconstructed from (e); (g) level-1 of the output hierarchy; (h) final output hair geometry.*

Abstract

We present an example-based approach to hair modeling because creating hairstyles either manually or through image-based acquisition is a costly and time-consuming process. We introduce a hierarchical hair synthesis framework that views a hairstyle both as a 3D vector field and a 2D arrangement of hair strands on the scalp. Since hair forms wisps, a hierarchical hair clustering algorithm has been developed for detecting wisps in example hairstyles. The coarsest level of the output hairstyle is synthesized using traditional 2D texture synthesis techniques. Synthesizing finer levels of the hierarchy is based on cluster oriented detail transfer. Finally, we compute a discrete tangent vector field from the synthesized hair at every level of the hierarchy to remove undesired inconsistencies among hair trajectories. Improved hair trajectories can be extracted from the vector field. Based on our automatic hair synthesis method, we have also developed simple user-controlled synthesis and editing techniques including feature-preserving combing as well as detail transfer between different hairstyles.

CR Categories: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—boundary representations I.3.7 [**Computer Graphics**]: Three-dimensional Graphics and Realism—color, shading, shadowing, and texture

Keywords: Hair Modeling, Texture Synthesis, Hair Clustering, Detail Transfer, Vector Fields

1 Introduction

Many graphics and internet applications, including film making (crowd simulation), game development and online virtual worlds, make heavy use of avatars whose hairstyles need to be modeled realistically with individuality. Designing visually realistic and pleasing hairstyles, either manually or through 3D image-based acquisition [Paris et al. 2004; Wei et al. 2005; Paris et al. 2008], is a costly and time-consuming process. This motivates an example-based methodology that creates novel hairstyles with reference to existing ones. Considering that hair roots are distributed over the 2D scalp surface, creating novel hairstyles from examples could be cast as a 2D texture synthesis problem. The fact, that hair strands with nearby roots share similar geometry and form wisps, indicates that the Markov random field assumption, commonly used in texture synthesis, also holds for most hairstyles. However, each hair strand is a 3D space curve with a long trajectory that is not solely determined by the location of the hair root. The tangents of all the hair strands form a 3D vector field whose consistency cannot be guaranteed by a 2D synthesis method. Likewise, the wisp structures of a hairstyle cannot be well preserved by indirect hair synthesis through the synthesis of the 3D hair tangent field. Therefore, a hair synthesis technique needs to simultaneously address both 2D and 3D aspects of a hairstyle.

There exist a few difficulties we need to overcome before novel hairstyles can be successfully synthesized. First, to employ 2D texture synthesis algorithms, we need to define a low-distortion parameterization of the scalp and the geometry of hair strands needs to be expressed as scalar channels of a 2D map. Second, characteristic wisp structures and the subtle geometric variations internal to the wisps are essential to the appearance of a hairstyle. A synthesized hairstyle needs to inherit these characteristics from the input hairstyle. Since each strand represents a long and complex trajectory, synthesizing the wisp structures of such long trajectories represents a significant challenge. Third, the 3D vector field formed by the tangents of hair strands describe their collective behavior in the 3D space. The spatial coherence of this vector field means when two hair trajectories pass through the same local region, the two trajectories tend to have similar tangent vectors within this region no matter how far apart their hair roots are. Observing this type of coherence during hair synthesis represents another challenge.

In this paper, we introduce a hierarchical hair synthesis framework that, given an input hairstyle, can create a novel one with a statistically similar spatial arrangement of hair strands and geometric details. It views a hairstyle both as a 2D arrangement of hair strands and a 3D vector field. Specifically, we have developed the following components. First, a low-distortion parameterization of the scalp and its surrounding hair volume is defined. Second, to support wisp oriented synthesis, an effective hierarchical clustering algorithm has been developed to detect clusters in the input hairstyle. Each cluster has a center strand as its representative. Third, traditional 2D texture synthesis is extended to coarsest-level hair synthesis. Synthesizing finer levels of hair is based on cluster oriented detail transfer. That is, only the differences between an input strand and its corresponding center strand are transferred to the synthesized hair. Finally, to enforce spatial coherence of tangent vectors, we compute a discrete vector field from the synthesized hair at every level of the hierarchy to remove undesired inconsistencies among hair trajectories. Improved hair trajectories with more coherent tangent vectors can be extracted from the vector field.

Based on the aforementioned hair synthesis framework, we have also developed simple mechanisms for user-controlled hair synthesis and editing, which can be performed either at a global scale by editing the hair geometry at the coarsest level of the hierarchy or at a local scale by transferring cluster-level geometric details from one hairstyle to another.

2 Related Work

Our work is made possible by recent progress on hair acquisition, hair modeling, texture synthesis and geometry synthesis.

Hair Acquisition. There has been significant progress on hair acquisition [Paris et al. 2004; Wei et al. 2005; Paris et al. 2008] from multiple camera views. An oriented filter bank with high angular resolution was adopted to detect hair orientation in [Paris et al. 2004]. Hand-held lights or cameras were used in [Paris et al. 2004; Wei et al. 2005] while triangulation was used in [Paris et al. 2008] to obtain more accurate 3D location in a multi-project, multi-camera setup. Hair acquisition is inherently a very hard problem because of the thin geometry of hair strands and numerous occlusions among them. It is almost impossible for a regular camera to directly acquire data from the interior of the hair volume. Existing methods typically acquire the outmost layer and interpolate inside the hair volume. The hair exemplars used as input to our synthesis algorithm were actually acquired from real hairstyles.

Hair Geometry Modeling. Synthetic hair modeling has a long history. Watanabe and Suenaga [1992] proposed a wisp model, extended by Chen et al. [1999]. In an integrated hair system by Daldegan et al. [1993], characteristic hair strands define the boundary of wisps. Exploiting the connections between vector fields and smooth hairstyles, interactive hairstyling systems have been developed by Hadap and Thalmann [2000] and Yu [2001]. A multiresolution hair modeling system based on the observation that adjacent hair strands tend to form clusters at multiple scales was developed by Kim and Neumann [2002]. Our synthesis algorithm is based on a hierarchical clustering of the input hairstyle. This clustering process can be regarded as the inverse process of the hierarchical modeling framework in [Kim and Neumann 2002]. An interactive technique based on a statistical wisp model and a physically based static deformation solver is proposed in [Choe and Ko 2005] to model a wide range of hairstyles. A survey of hair modeling techniques can be found in [Ward et al. 2007]. Note that none of these techniques relies on captured hairstyles to generate novel ones.

Texture Synthesis. This paper was partially inspired by the recent success in texture synthesis [Bonet 1997; Efros and Leung 1999; Wei and Levoy 2000; Ashikhmin 2001; Liang et al. 2001; Efros and Freeman 2001; Hertzmann et al. 2001; Kwatra et al. 2003; Kwatra et al. 2005; Lefebvre and Hoppe 2005; Lefebvre and Hoppe 2006; Han et al. 2008]. A representative approach [Efros and Leung 1999; Wei and Levoy 2000] models textures as Markov Random Fields and generates novel textures by non-parametric conditional pixel-based sampling. This approach was further generalized to non-parametric patch-based sampling [Efros and Freeman 2001; Liang et al. 2001]. Kwatra et al. [2005] considers patch-based texture synthesis as an energy optimization problem and achieved robust results. A GPU-based parallel synthesis algorithm was developed in [Lefebvre and Hoppe 2005], which was further extended to appearance-space synthesis in [Lefebvre and Hoppe 2006]. A multiscale texture synthesis method capable of infinite zoom has been proposed in [Han et al. 2008]. There has also been much work on generalizing 2D texture synthesis to meshes [Praun et al. 2000; Turk 2001; Wei and Levoy 2001] and volumes [Kopf et al. 2007; Takayama et al. 2008].

As discussed earlier, a hairstyle should be regarded both as a 2D spatial arrangement of hair strands on the scalp and a 3D vector field. Existing texture synthesis methods are not directly applicable to hair because either 2D or 3D texture synthesis addresses only one of the two aspects of hair. There exists much work on generalizing texture synthesis to curves [Hertzmann et al. 2002] and geometric synthesis [Bhat et al. 2004; Lai et al. 2005; Lagae et al. 2005; Zhou et al. 2006; Zhou et al. 2007]. Nevertheless, none of them is applicable to a volumetric packing of thin curves, such as hair.

3 Scalp Space

The scalp is a curved surface that is to a certain extent similar to a hemisphere. To facilitate hair strand comparison and hairstyle synthesis, we define a 3D parameterization, named the scalp space, for the curved hair volume bounded by the scalp from below. A 2D parameterization of the scalp surface can be easily extracted by simply discarding the third parameter. At every point on the scalp and in the hair volume, we also define a local frame taking into account the local tangent space on the scalp.

Without loss of generality, we can assume that the scalp surface is similar to the upper half of a unit sphere. The world coordinate frame is originated at the center of the hemisphere. Its y-axis points upward. Given a world space point $P = (x, y, z)$, its scalp space coordinates (u, v, w) are defined as follows:

$$\begin{aligned} u &= \arccos \frac{\hat{x}}{\sqrt{\hat{x}^2 + (\hat{y}+1)^2}} \\ v &= \arccos \frac{\hat{z}}{\sqrt{\hat{z}^2 + (\hat{y}+1)^2}} \\ w &= \sqrt{x^2+y^2+z^2} - d(\hat{x}, \hat{y}, \hat{z}) \end{aligned} \tag{1}$$

where $(\hat{x}, \hat{y}, \hat{z})$ is the spherical projection of (x, y, z) onto the unit sphere, $d(\hat{x}, \hat{y}, \hat{z})$ is the distance from the center of the hemisphere to the scalp along the direction defined by $(\hat{x}, \hat{y}, \hat{z})$.

As shown in Figure 2, the parameter u can be intuitively interpreted as the angle between the x-axis and the projection of S_0P onto the XY-plane where $S_0 = (0, -1, 0)$. The parameter v can be defined similarly with respect to the YZ-plane and z-axis. w is simply the height above the scalp.

The backward mapping can be derived easily from Eq. (1):

$$\begin{aligned} x &= \rho h \cot u \\ y &= \rho(h-1) \\ z &= \rho h \cot v \end{aligned} \tag{2}$$

where $h = \frac{2}{\cot^2 u + \cot^2 v + 1}$ and $\rho = w + d(h \cot u, h-1, h \cot v)$.

Unlike the parameterization defined by longitude and latitude, our parameterization of the scalp only has a single singularity at $S_0 =$

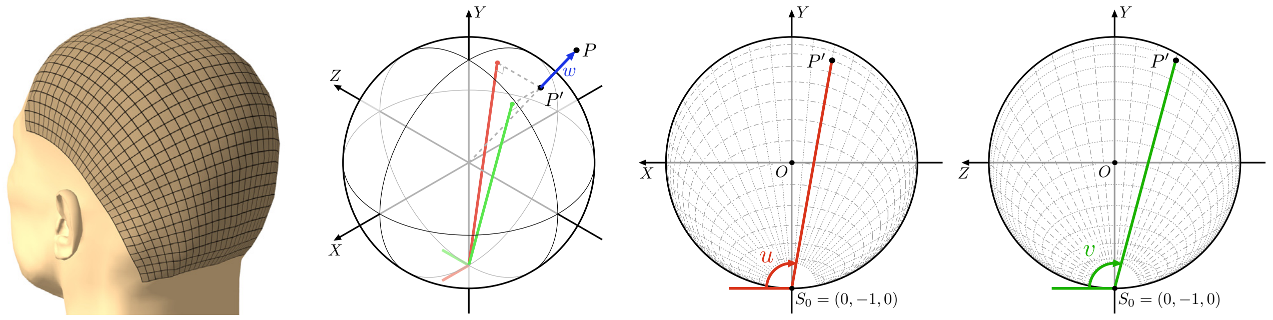


Figure 2: *3D Scalp Space Parameterization. Given a world space point $P(x, y, z)$ and its spherical projection P' on the unit sphere, its scalp space coordinates (u, v, w) has an intuitive meaning: u and v can be interpreted as two angles ranging from 0 to π as shown in the two rightmost figures.*

$(0, -1, 0)$. Distortion introduced by our parameterization becomes larger as y decreases. In practice, however, we can always map the hair roots to the upper hemisphere where the distortion is acceptable for our application.

We also define a local tangent frame at every point on the scalp surface. To compute the tangent and binormal vector within the frame, we first compute a rotation matrix, $\mathbf{R}$, that represents the rotation from the y-axis to the local surface normal. Then, the local tangent and binormal are defined by rotating the x- and z-axis respectively using $\mathbf{R}$. This definition can be extended to any point in the hair volume. The local frame at such a point shares the same coordinate axes with the point on the scalp that has the same u and v.

4 Hierarchical Hair Clustering

To reproduce characteristic hierarchical wisp structures during example-based hairstyle synthesis, it is crucial to identify such structures for the example hairstyles. Inspired by the triangle clustering algorithm in [Cohen-Steiner et al. 2004], we introduce a variational hair clustering algorithm which automatically divides the original set of hair strands into k clusters and computes a *center strand* for each cluster. The set of these k clusters is called a k-partition. With this algorithm, building the full hierarchy follows a straightforward process: at each level of the hierarchy, we divide the hair strands at the current level into clusters and let the center strands be the hair strands in the next (coarser) level.

Our hair clustering follows the general approach of Lloyd's algorithm (a.k.a. *k-means clustering*) [Lloyd 1982], which is guaranteed to converge. Lloyd's algorithm alternately repeats the following two phases: *partitioning* and *fitting*. For the first phase, we design a cluster growing algorithm that simultaneously expands all the clusters from their seeding strands to cover the entire set of strands. It guarantees that strands within the same cluster are connected according to a predefined neighborhood structure. In the second phase, we compute for each cluster an optimal local representative, the center strand.

4.1 Energy Function

Defining an appropriate energy function is a key ingredient in clustering. Given two hair strands γ_a and γ_b, we first discretize each of them into n_s sample vertices. Let $\gamma_a(l)$ and $\gamma_b(l)$ denote the coordinates of the l-th vertex (from root to tip) in the local frame defined at the root of γ_a and γ_b, respectively. Then we define the $\mathcal{L}^2$ distance between them as $\mathcal{L}^2(\gamma_a, \gamma_b) = \sum_{l=1}^{n_s} \|\gamma_a(l) - \gamma_b(l)\|^2$. Note that in this equation we do not add more weights to the hair roots, because we would like to emphasize the similarity between "more visible" portions of the hair strands. We refer to the input set of hair strands as $\mathcal{S}$, its k clusters as $\mathcal{S}_i$, and their current respective center strand as $\bar{\gamma}_i$. The overall energy function for hair clustering is defined as

$$\mathcal{E}(\mathcal{S}) = \sum_{i=1}^{k} \sum_{j=1}^{|\mathcal{S}_i|} \mathcal{L}^2(\gamma_j^i, \bar{\gamma}_i), \tag{3}$$

where γ_j^i represents the j-th strand in the i-th cluster.

4.2 Partitioning

Knowing a fixed set of center strands, we wish to update the partition while minimizing the energy function in (3). For the i-th cluster of the previous partition, we first locate a *seed strand* $\gamma_1^i \in \mathcal{S}_i$ that is most similar to its associated center strand, i.e., $\gamma_1^i = \arg\min_{\gamma_j \in \mathcal{S}_i} \mathcal{L}^2(\gamma_j, \bar{\gamma}_i)$. In the very first iteration, the partitioning phase picks k strands at random, and each of these strands is designated a center strand as well as a seed strand. The assignments of all the other strands are initially set to null. In order to cluster together only strands that are similar to the center strand, for each seed strand γ_1^i, we insert every strand γ_j in its neighborhood into a global priority queue, with a priority inversely proportional to the distance to the corresponding center strand, $\mathcal{L}^2(\gamma_j, \bar{\gamma}_i)$, and we further post a tag indicating the cluster label i of the center strand it is being tested with. The neighborhood of a strand is acquired by Delaunay triangulation of all hair roots.

The region-growing process proceeds by repeatedly popping hair strands with the highest priority until the queue is empty. For each popped strand, we check its cluster assignment. If it has already been assigned to a cluster, we do nothing and skip to the next strand in the queue; otherwise, we assign it to the cluster indicated by its tag, and push its unlabeled neighboring strands into the queue along with the same tag. When the priority queue has been emptied, each strand has been assigned to a cluster. Notice that this process ensures connected and non-overlapping clusters as required, and that it has a low computational complexity, $N \log N$, where N is the total number of strands.

4.3 Fitting

Once we have obtained a new partition, we wish to update the center strand of every cluster in order for it to be the best representative. For the $\mathcal{L}^2$ distance metric, the center strand that minimizes the energy function in (3) is simply the average of the hair strands in each cluster.

4.4 Adaptive Clustering

Usually it is inconvenient for the user to decide the proper number of clusters for a given hairstyle. Therefore we have made a simple extension, called *adaptive clustering*, that allows the algorithm to determine an appropriate number of clusters based on a user-given error threshold ϵ. The adaptive clustering algorithm aims to iteratively find a k-partition for a given set of strands $\mathcal{S}$ so that for any strand γ_j in $\mathcal{S}$ and its associated center strand $\bar{\gamma}_i$, the distance between them, $\mathcal{L}^2(\gamma_j, \bar{\gamma}_i)$, is always less than ϵ. The algorithm starts by performing fixed-k clustering on $\mathcal{S}$ with a relatively small initial number of clusters, k_0. In each resulting cluster, if the distance from its center strand to certain strands in the cluster is larger than ϵ, the strand with the largest distance to the center strand is set to be the seed of a new cluster.

When building the full hierarchy for a given hairstyle, we apply adaptive clustering only on the finest level. The number of clusters in a coarser level is set to be one fourth of that in the finer level.

Figure 3 shows the overall energy $\mathcal{E}(\mathcal{S})$ as a function of the number of iterations. As can be seen, a few iterations suffice to reach a much reduced value. Due to the region growing nature that guarantees the connectedness of every resulting cluster, there is no theoretical proof that our iterative clustering always converges. However, convergence has been achieved in all our experiments.

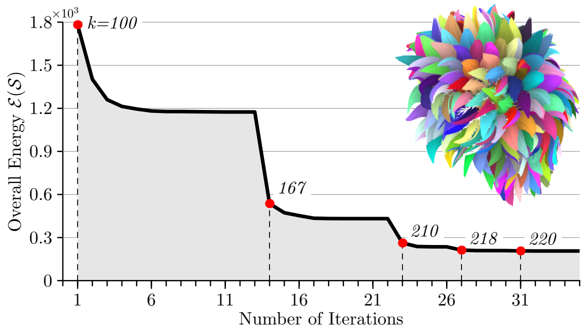


Figure 3: *A hairstyle with 40,000 strands is divided into 220 clusters using our adaptive clustering algorithm. The curve shows the overall energy $\mathcal{E}(\mathcal{S})$ as a function of the number of iterations; red dots indicate the cluster splitting stages; the number of clusters (k) is shown in italics next to the red dots.*

5 Hierarchical Cluster-Based Synthesis

The hair clustering stage generates a full hierarchy for the input hairstyle from which we can synthesize novel hairstyles. The synthesis starts from the coarsest hierarchical level where the synthesized hairstyle models the global hair flow. We represent the geometry of each strand using a multi-dimensional feature vector and utilize a 2D texture synthesis algorithm to synthesize a 2D map of these feature vectors (Section 5.1). Once we have the coarsest level of the output hairstyle, we synthesize finer levels by progressively adding more geometric details to the global hair flow. Since we would like to preserve large-scale hair geometry synthesized in coarser levels while adding more details, our algorithm only transfers displacements between hair strands and the center strand of the cluster they belong to (Section 5.2).

In hairstyle synthesis, the meaning of spatial coherence is two fold. First, when hair roots are close to each other on the scalp surface, corresponding hair strands tend to form wisps and have similar geometry. Second, every hair strand has a curved trajectory in the hair volume and spatially close portions of the trajectories tend to share similar tangent vectors. The aforementioned cluster based synthesis only considers the first type of coherence. We improve the second type of coherence with a 3D vector field representing the tangents of hair strands. After synthesizing each level, we compute a discrete vector field based on the tangent vectors of the hair strands on that level. Updated hair strands with improved consistency and continuity are generated by tracing this vector field (Section 5.3). The details of these synthesis steps are presented in the following subsections.

5.1 Feature Map Synthesis

Hair synthesis at the coarsest level of the hierarchy is reduced to classical 2D texture synthesis. In hair clustering, we have discretized a hair strand into a fixed number (n_s) of sample vertices and evaluate each vertex's coordinates in the local frame defined at the root of the hair strand. Concatenating the 3D coordinates of all these hair vertices together, we obtain a $3n_s$-dimensional vector. We further perform Principal Components Analysis (PCA) on all vectors of hair geometry and project them to a n_f-dimensional subspace, yielding what we call the feature vectors. Note that PCA projection has been previously used in texture synthesis [Hertzmann et al. 2001; Liang et al. 2001; Lefebvre and Hoppe 2006] but not in hair modeling. In practice, we have found that $n_f = 16$ is sufficient for all the hairstyles we tested.

With the definition of hair feature vectors, we can now generate a 2D feature map $\mathcal{F}(\mathcal{S})$ for any given hair set $\mathcal{S}$ using our (u, v) parameterization of the scalp surface discussed in Section 3. Since the hair roots are irregularly distributed on the scalp surface, we need to compute interpolated feature vectors at regularly spaced grid points in the feature map. Once we have the input feature map, theoretically we could use any existing 2D texture synthesis method to generate a new feature map for the output hairstyle. In the current implementation, we use texture optimization [Kwatra et al. 2005] for its high quality.

Once we have a synthesized feature map for the output hairstyle, we can reconstruct the geometry of the hair strands from their PCA coefficients. We first randomly generate hair roots within a user specified scalp region, and apply the relaxation procedure in [Turk 2001] to make the hair roots more uniformly distributed. Then for each hair root, we find a spatially closest feature vector from the synthesized feature map and reconstruct the geometry. The number of generated strands is set to roughly match the spatial density of hair roots in the same level of the input hierarchy.

5.2 Cluster Oriented Detail Transfer

To perform hierarchical cluster based synthesis, we need to generate a hierarchy of clusters for the *output* hairstyle. We adopt the top-down approach in [Kim and Neumann 2002] to build such a hierarchy. The basic idea is to consider every hair strand at a coarser level as a center strand and assign those hair strands at the finer level to the closest strand at the coarser level.

Wisp structures and the subtle geometric variations internal to the wisps are essential characteristics of a hairstyle. A synthesized hairstyle needs to inherit these characteristics from the input hairstyle. Thus, we chose to hierarchically synthesize geometric details for the synthesized hairstyle on a per-cluster basis. In our hierarchical synthesis, a strand at a coarser level serves as the center strand of a cluster at the next finer level. When synthesizing a level except the coarsest in the output hierarchy, we find for each output cluster at the current level a corresponding input cluster at the same level of the input hierarchy. This is achieved by locating the input cluster whose center strand is most similar to the output cluster's center strand, which is inherited from the coarser level. This search is accelerated using the PCA-based feature vectors as in Section 5.1, with two modifications. First, a per-strand-aligned local tangent frame is obtained for each center strand $\bar{\gamma}_i$ by rotating the local frame defined in Section 3 around its normal, so that the tangent vector in the new frame is aligned with the *average tangent vector* of $\bar{\gamma}_i$ projected onto the scalp tangent plane. The average tangent vector of any strand γ is defined as

$$\mathbf{v}_g(\gamma) = \frac{\sum_{i=2}^{n_s}(\gamma(i) - \gamma(1))}{\| \sum_{i=2}^{n_s}(\gamma(i) - \gamma(1))\|}. \quad (4)$$

The second modification is that the input vectors of PCA contain *both* center strands of the input and output hairstyles, measured under the aforementioned per-strand local frames.

Note that these modifications enable similarity measurement within per-strand-aligned local frames to discount any differences in the global orientation of the strands. This is important to the displacement-based detail transfer.

Given a pair of corresponding clusters, $\mathcal{S}_j^{in}$ and $\mathcal{S}_i^{out}$, at the same level in the input and output hierarchies, respectively, we superpose the extracted geometric details from $\mathcal{S}_j^{in}$ onto the center strand of $\mathcal{S}_i^{out}$ to yield the final geometry of the strands in $\mathcal{S}_i^{out}$. The center strand of $\mathcal{S}_i^{out}$ thus supplies the base geometry. Note that initially all strands except the center strand of the output cluster $\mathcal{S}_i^{out}$ have no geometric information but their root locations.

Displacement Definition. Given two discretized hair strands γ_a and γ_b, $\gamma_a(i)$ and $\gamma_b(i)$ denote the world space coordinates of the i-th vertex of γ_a and γ_b respectively. The displacement of γ_a with respect to γ_b, is evaluated on every vertex (sample point). The displacement of γ_a from γ_b can be divided into two components $\mathcal{D}_\alpha(\gamma_a, \gamma_b)$ and $\mathcal{D}_\beta(\gamma_a, \gamma_b)$ which we will define separately. We use the term $\mathcal{D}(\gamma_a, \gamma_b, i)$ to denote the displacement of γ_a's i-th vertex.

The first component, $\mathcal{D}_\alpha(\gamma_a, \gamma_b)$, describes the *relative* displacement between two hair strands in a *per-strand* local frame. The computation can be imagined as first making γ_a's root coincidental with γ_b's root, then computing per-vertex displacements in the local frame at γ_b's root location. More formally, we have

$$\mathcal{D}_\alpha(\gamma_a, \gamma_b, i) = \mathbf{M}_{nbt}^T(\gamma_b)[(\gamma_a(i) - \gamma_b(i)) - (\gamma_a(1) - \gamma_b(1))], \quad (5)$$

where $\mathbf{M}_{nbt}^T(\gamma_b)$ is the matrix that transforms world space coordinates to the local frame at γ_b's root location.

The second component, $\mathcal{D}_\beta(\gamma_a, \gamma_b)$, represents the *absolute* displacement in *per-vertex* local frames. To compute $\mathcal{D}_\beta(\gamma_a, \gamma_b)$, we need to first compute the per-vertex local frame defined by the normal, bi-normal and tangent along γ_b. [Bloomenthal 1990] provides a method to compute such local frames along a space curve. Then we can compute $\mathcal{D}_\beta(\gamma_a, \gamma_b)$ as the per-vertex displacements represented in the per-vertex local frames,

$$\mathcal{D}_\beta(\gamma_a, \gamma_b, i) = \mathbf{M}_{nbt}^T(\gamma_b(i))(\gamma_a(i) - \gamma_b(i)), \quad (6)$$

where $\mathbf{M}_{nbt}^T(\gamma_b(i))$ is the local frame defined by the normal, binormal and tangent at the i-th vertex of γ_b.

Displacement Transfer. Displacement transfer works as follows. For each output strand γ^{out} in cluster $\mathcal{S}^{out}$, we first compute the displacement between its root position and the root of the center strand $\bar{\gamma}^{out}$ of $\mathcal{S}^{out}$, and then find an input strand γ^{in} with the most similar displacement with respect to the root of the center strand $\bar{\gamma}^{in}$ of $\mathcal{S}^{in}$. Finally, we construct γ^{out} by linearly interpolating two different estimations using spatially varying interpolation coefficients.

$$\gamma^{out}(i) = \frac{n_s - i}{n_s - 1}\gamma_\alpha^{out}(i) + \frac{i-1}{n_s - 1}\gamma_\beta^{out}(i), \quad i = 1, \ldots, n_s \quad (7)$$

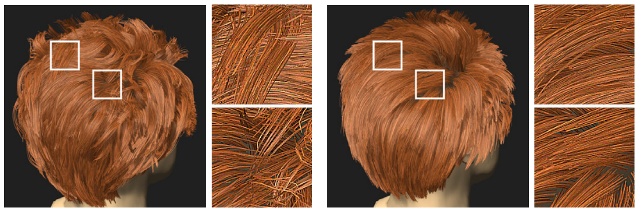


Figure 4: *A comparison of synthesized hairstyles without (left) and with (right) vector field based refinement. The tangent field of the hairstyle on the right is more spatially coherent. The input hairstyle can be found in the middle of the third column in Figure 7.*

where

$$\begin{aligned} \gamma_\alpha^{out}(i) &= \bar{\gamma}^{out}(i) + \mathbf{M}_{nbt}(\bar{\gamma}^{out})\mathcal{D}_\alpha(\gamma^{in}, \bar{\gamma}^{in}, i) + \\ &\quad \gamma^{out}(1) - \bar{\gamma}^{out}(1) \\ \gamma_\beta^{out}(i) &= \bar{\gamma}^{out}(i) + \mathbf{M}_{nbt}(\bar{\gamma}^{out}(i))\mathcal{D}_\beta(\gamma^{in}, \bar{\gamma}^{in}, i). \end{aligned} \quad (8)$$

Here $\gamma_\alpha^{out}(1)$ guarantees the synthesized strand γ^{out} is rooted at the prescribed location on the output scalp while $\gamma_\beta^{out}(n_s)$ ensures that the relative position between the tips of γ^{in} and $\bar{\gamma}^{in}$ is actually preserved between γ^{out} and $\bar{\gamma}^{out}$.

5.3 3D Vector Field Based Hair Refinement

The feature vector for each hair strand is stored spatially at its root location while the strand represents a long and complex trajectory and thus affects the appearance everywhere along this trajectory. Furthermore, some hair strands are "coherent" across a distance greater than that can be reflected in a feature map. For instance, two hair strands, whose roots are not quite close to each other in the feature map, could actually be "well aligned" in the 3D space to form a continuous and smooth path which is visually longer than any of them, as illustrated in Figure 5. To improve this type of spatial coherence, we introduce a refining process that first computes a 3D vector field from initially synthesized hair geometry, then traces updated hair strands from the vector field to replace the original strands.

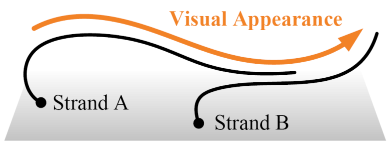


Figure 5: *Two spatially continuous hair strands could be viewed as one longer hair strand.*

We define a discrete vector field using a 3D grid embedded in the scalp space described in Section 3. Let (i, j, k) denote the indices of a grid cell located at the scalp space coordinates (u_i, v_j, w_k); $d\mathbf{s}$ denotes a small segment of any hair strand; $\mathbf{T}(d\mathbf{s})$ denotes the tangent vector of $d\mathbf{s}$; $C(i, j, k)$ denotes the set of hair segments that pass through the grid cell (i, j, k). The vector field is then computed iteratively as follows.

$$\bar{\mathbf{v}}^t(i, j, k) = \frac{\sum_{d\mathbf{s} \in C(i,j,k)} \omega^t(\mathbf{T}(d\mathbf{s}))\mathbf{T}(d\mathbf{s})}{\| \sum_{d\mathbf{s} \in C(i,j,k)} \omega^t(\mathbf{T}(d\mathbf{s}))\mathbf{T}(d\mathbf{s})\|}, \quad (9)$$

where $\omega^t(\mathbf{v}) = 0.5(\mathbf{v} \cdot \bar{\mathbf{v}}^{t-1}(i, j, k) + 1)$ and $\bar{\mathbf{v}}^0(i, j, k) = \mathbf{0}$. In case there are no hair segments passing through a cell, the estimated vector there is also set to $\mathbf{0}$. Note that a hair strand can be parameterized using its arc length from the root and the tangent vector of any hair segment along the strand can be unambiguously defined. Therefore, it is not necessary to employ structure tensors as in [Paris et al. 2008].

The iteratively changing weighting scheme is aimed at outlier suppression. It ensures that within each grid cell, the more different the direction of a hair segment is from the average vector, the less it will affect the final estimated vector. As we know, given a set of inconsistent estimations of the vector within a grid cell, their average vector is actually the least-squares solution. However, least-squares solutions can be easily corrupted by outliers. It has been shown that solutions from iteratively re-weighted least squares can asymptotically approach the optimal solution computed by a robust estimator [Hampel et al. 1986]. In practice, the above estimation terminates after two iterations because we found that more iterations did not improve the visual results any more.

The creation of updated hair strands from the vector field is almost the same as in previous work [Paris et al. 2004; Wei et al. 2005; Paris et al. 2008]. Each hair strand is a streamline of the vector field integrated using the Euler method. The integration starts from its original root position. The actual vector field used for integration is an interpolated version of the previously computed discrete field. For a location not coincidental with the center of any of the grid cells, the vector there is trilinearly interpolated from eight nearby cells. The integration terminates when either the strand has reached a predefined length, or it has reached a location where the interpolated vector $\bar{\mathbf{v}} = 0$. The predefined length is randomly sampled from the input hairstyle in a region on the scalp surrounding the root of the output strand.

Overall, this vector field based refinement improves the consistency and continuity among nearby hair trajectories (Figure 4). Note that a vector field whose grid spacing is too large can smooth out small geometric features in the hair. An unnecessarily small grid spacing, on the other side, not only increases the time complexity, but may also leave artifacts unaffected by the refinement. Therefore, one should choose a reasonably small grid size according to the feature size of the input hairstyle (some typical values of the grid size are given in the next section).

6 Results and Discussion

We have extensively experimented with our hair synthesis algorithm. In Figures 1 and 6, we show several synthesized hairstyles side-by-side with example hairstyles. All the synthesis results in these figures were achieved without any user intervention. Note that most of the input hairstyles were acquired using image-based methods while the others were manually created. The input hairstyles in Figure 1 and 6(b) went through an initial cleanup stage as described in the supplemental materials.

Implementation. Although a few parameters in our pipeline are adjustable, we have found their typical values (as shown in Table 1) that work for all the examples we have tested. Two parameters need special attention during hair synthesis. First, using more hair hierarchy levels has the effect of extracting higher-level trends from local details and better grasp the spatial characteristics of global hair flows. For simple hairstyles like the spiky one in Figure 6(d), two scales can achieve good results. For all the other hairstyles in the paper, we use three scales because of their complexity. Using more than three scales does not visually improve the results in practice. Second, the choice of the grid spacing of the 3D vector field should match the complexity of hair geometry in the respective hierarchy level. In our implementation, we use 0.02 for the coarsest level and set the spacing in a finer level to be half of that in the coarser level. Also, for the parameters in texture optimization, we simply use the same settings as in [Kwatra et al. 2005]. Hairstyles in this paper are rendered in real time as anti-aliased polylines with single scattering computed using [Marschner et al. 2003], diffuse and ambient components computed using [Kajiya and Kay 1989] and shadows computed using [Yuksel and Keyser 2008].

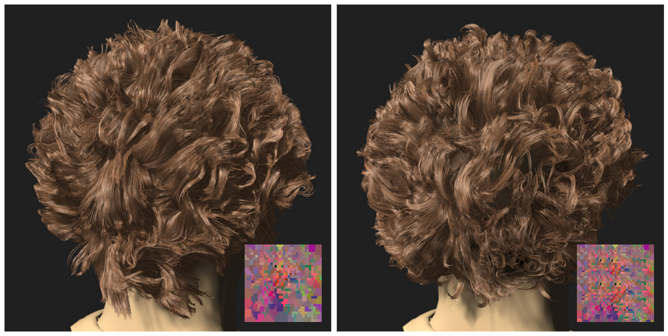
Symbol	*Description*	*Typical values*
n_h	number of hair hierarchy levels	2 to 3
k_0	initial number of clusters	200
ϵ	fitting error threshold	0.1 to 0.3
n_s	number of sample points	64
n_f	dimensionality of PCA subspace	16
Δ_{2D}	feature map grid spacing	0.02
Δ_{3D}	vector field grid spacing	0.02
δ	integration step for hair growing	0.02

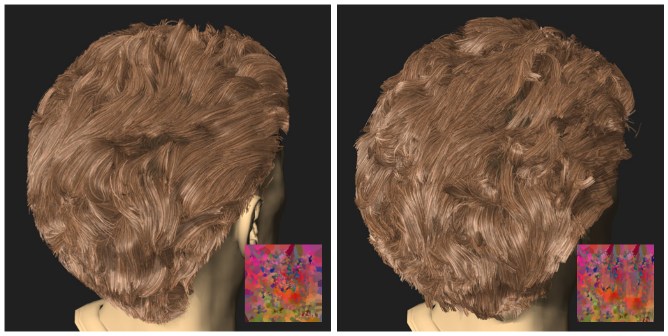
Table 1: *Adjustable parameters and their typical values. Note that we assume the radius of the virtual head is roughly 1.*

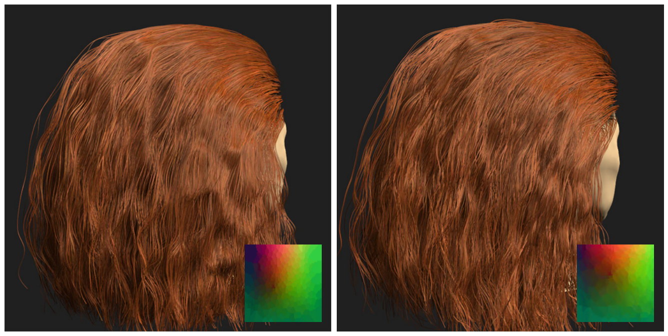
Quality and Performance. Our automatically synthesized hairstyles are able to capture the essence of the spatial hair arrangement as well as cluster oriented characteristics of the original examples. Meanwhile, our method ensures sufficient individuality. For a hairstyle with about 100,000 strands, the overall synthesis time is usually less than two minutes on a Pentium 4 3.0GHz processor with 2GB memory. Hierarchical clustering accounts for about 50 percent of the time while texture optimization for the feature map at the coarsest level and 3D vector field based refinement each takes about 20 percent. Displacement transfer is rather fast. Note that due to our hierarchical hair clustering, 2D texture synthesis is performed only at a coarse resolution (typically lower than 100×100), which significantly reduces the overall time complexity.

User-Controlled Synthesis. Our framework also allows users to interactively control synthesis at both the global scale and per-cluster scale to yield interesting effects. Based on hierarchical clustering of hair strands, one can, for example, either modify the coarsest level of the hair geometry generated using a feature map, or directly impose an arbitrary coarsest level, to control the global flow of the synthesized hairstyle. We have implemented a stroke-based GUI tool that emulates the function of a real comb and allows the user to deform hair strands intuitively. Note that such hair combing tools are available in many existing modeling softwares, such as Blender and 3D Studio Max. Nevertheless, our combing tool enables detail-preserving editing by first applying detail-preserving deformations to the strands in the coarsest level and then transferring original finer-level details onto the edited coarsest level. Detail-preserving deformation of hair strands in the coarsest level is achieved with the technique described in “Gradient Domain Editing of Deforming Mesh Sequences” [Xu et al. 2007] (section 3.1). The user can also transfer detailed geometric variations from one hairstyle to another. This is done by retaining the original coarsest level of the second hairstyle and transferring finer-level details from the first one. To ensure a thorough high-frequency detail transfer, we further smooth the geometry of the individual hair strands at the coarsest level of the first hairstyle. Examples of these types of user-controlled synthesis can be found in Figure 7, where compelling detail transfer results have been achieved.

Comparison with Conventional 2D/3D Synthesis. We have compared our synthesis method with conventional 2D and 3D synthesis algorithms. Conventional 2D synthesis would directly perform feature map synthesis at the finest level of the hierarchy and reconstructs the geometry of the hair strands using the PCA coefficients from the synthesized feature map. A comparison between pure 2D synthesis and our method can be found in Figure 8 (Top). It can be seen that conventional 2D synthesis cannot well preserve


(a) Puffy


(b) Wavy


(c) Curly

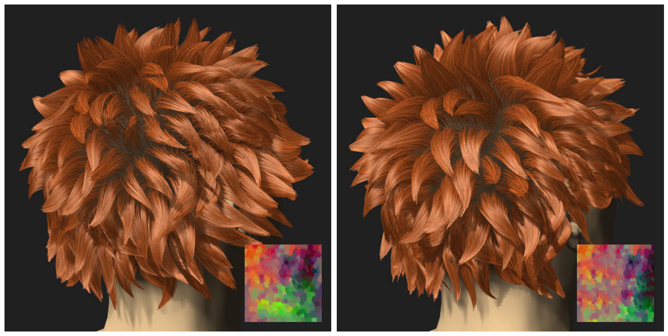

(d) Spiky

Figure 6: *Hair synthesis results from various examples. In each group, the input is on the left while the output on the right. The inset shows the respective feature map at the coarsest level. In addition to synthesizing hairstyles acquired using real images ((a) and (b)), our algorithm can also synthesize manually created ones ((c) and (d)).*

the wisp structures and their spatial arrangement in the original input hairstyle.

We have also experimented with conventional 3D synthesis where the tangent field of the input hairstyle is taken as an input example for 3D vector field synthesis. Streamlines traced from the synthesized 3D vector field collectively form the synthesized hairstyle. We generalize texture optimization [Kwatra et al. 2005] to 3D vector field synthesis by using 3D neighborhoods and considering the three coordinates of the vectors as color channels. Since hair strands are always grown from the scalp, we organize the 3D hair volume as a set of 2D curved slices using our hair volume parameterization and synthesize the vectors in the slice closest to the scalp first. A comparison between this type of 3D vector field synthesis and our method can be found in Figure 8 (Bottom). It can be seen that conventional 3D synthesis cannot preserve the wisp structures and other large-scale features either. In addition, it is roughly one to two orders of magnitude slower than our method.

Limitations. Our method is not directly applicable to draped long hairstyles. When performing the 3D vector field based refinement for such hairstyles, our scalp space parameterization would not be appropriate any more since the projection of certain hair vertices onto the unit sphere will approach the singularity of our parameterization at the south pole. One possible solution is to use a regular 3D grid in the world space rather than in the curved scalp space for any vector field oriented computation. Another limitation is that our current implementation for detail transfer between different hairstyles would introduce scale distortions in the transferred details when hair length of the involved hairstyles has significant difference. This problem can be potentially fixed using example-based curve synthesis [Hertzmann et al. 2002] to generate hair strands with both desired style and length. Finally, due to the trade-off of the vector field grid spacing (as described in Sec. 5.3), it would be hard for our method to achieve good results on hairstyles with tiny loops or dreadlocks.

7 Conclusions and Future Work

We have presented a hierarchical hair synthesis framework that, given an input hairstyle, can create a novel one with a statistically similar spatial arrangement of hair strands and geometric details. It has effective algorithmic components that efficiently support both 2D and 3D aspects of hairstyles. Based on our automatic hair synthesis method, we have also developed simple mechanisms for user-controlled hair synthesis and editing. We have demonstrated the quality of both automatic and user-controlled hair synthesis.

An intriguing item worth further investigation is that our hierarchical hair clustering algorithm might be able to bridge the gap between manual creation of hair models and image-based methods so that captured hair data could be clustered first to facilitate any follow-up editing performed by artists. Creating novel hairstyles by editing captured ones could dramatically reduce hair modeling costs. We have already partially demonstrated this possibility through our combing tool. A closely related research topic is how to simultaneously exploit both the 2D and 3D nature of hairstyles to conveniently and effectively perform hair editing.

Acknowledgements

We would like to thank Cem Yuksel, Sylvain Paris and colleagues for sharing their geometric hair data, Matt Scott for video dubbing, Steve Lin and Daisy Hao for help on writing, and the anonymous reviewers for their valuable suggestions. Kun Zhou was partially supported by NSFC (No. 60825201), the 973 program of China (No. 2009CB320801) and NVIDIA.

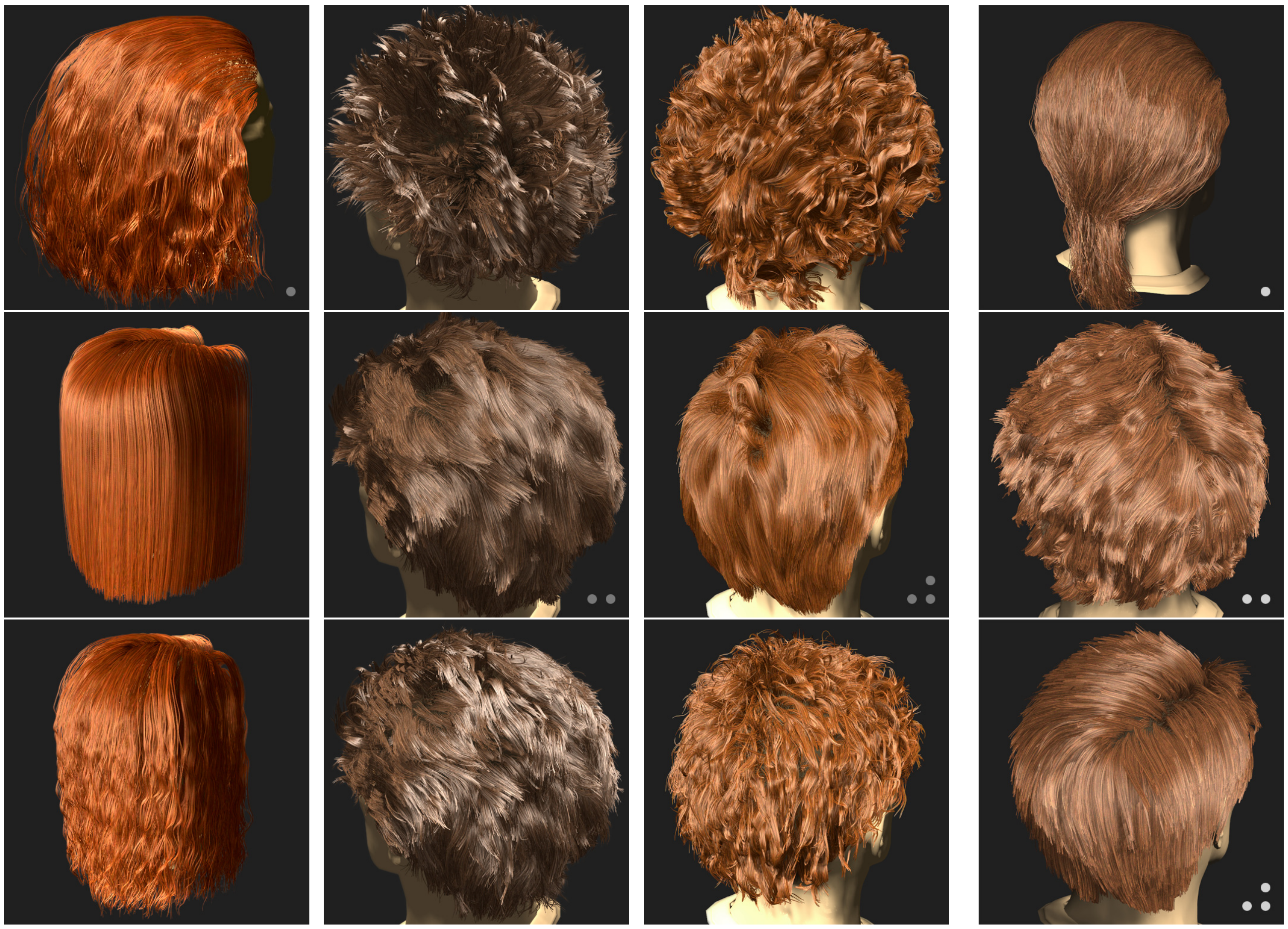


Figure 7: *Hair detail transfer (in (a),(b),(c)) and combing (in (d)) results. In the first three columns, the details from the hairstyles in the top row are transferred to the coarsest level of the hairstyles in the middle row to produce the final results in the bottom. In the rightmost column, three different hairstyles (as marked by dots in the first three columns) are edited using our detail-preserving comb.*

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, 217–226.

BHAT, P., INGRAM, S., AND TURK, G. 2004. Geometric texture synthesis by examples. In *Eurographics Symposium on Geometry Processing*, 41–44.

BLOOMENTHAL, J. 1990. Calculation of reference frames along a space curve. *Graphics Gems*, 567–571.

BONET, J. D. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proc. of SIGGRAPH*, 361–368.

CHEN, L.-H., SAEYOR, S., DOHI, H., AND ISHIZUKA, M. 1999. A system of 3D hair style synthesis based on the wisp model. *The Visual Computer 15*, 4, 159–170.

CHOE, B., AND KO, H.-S. 2005. A statistical wisp model and pseudophysical approaches for interactive hairstyle generation. *IEEE Transactions on Visualization and Computer Graphics 11*, 2, 160–170.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM Trans. Graph. 23*, 3, 905–914.

DALDEGAN, A., THALMANN, N., KURIHARA, T., AND THALMANN, D. 1993. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum (Eurographics'93) 12*, 3, 211–221.

EFROS, A., AND FREEMAN, W. 2001. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01*, 341–346.

EFROS, A., AND LEUNG, T. 1999. Texture synthesis by non-parametric sampling. In *ICCV '99*, 1033–1038.

HADAP, S., AND MAGNENAT-THALMANN, N. 2000. Interactive hair styler based on fluid flow. In *Computer Animation and Simulation 2000. Proceedings of the 11th Eurographics Workshop*.

HAMPEL, F., ROUSSEEUW, P., RONCHETTI, E., AND STAHEL, W. 1986. *Robust Statistics*. John Wiley & Sons, New York.

HAN, C., RISSER, E., RAMAMOORTHI, R., AND GRINSPUN, E. 2008. Multiscale texture synthesis. *ACM Transactions on Graphics 27*, 3, 51.

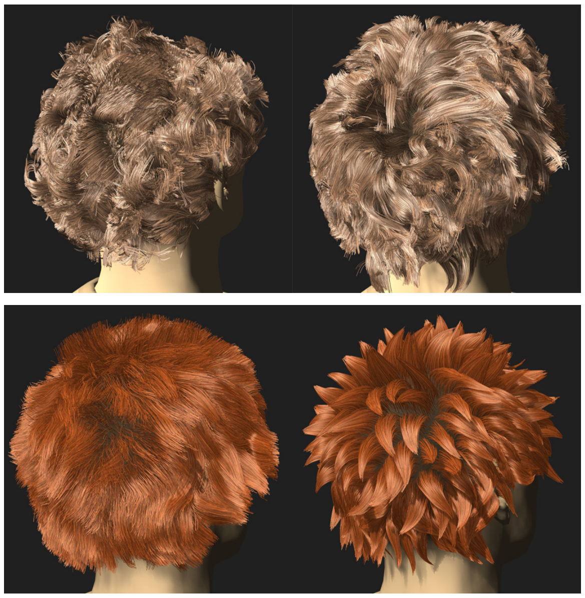


Figure 8: *Two comparisons with alternative methods. Top row: synthesized tangled hairstyles using conventional 2D texture synthesis (left) and our method (right). Bottom row: synthesized spiky hairstyles using 3D vector field synthesis (left) and our method (right). The original tangled and spiky hairstyles can be found in Figures 1 and 6 respectively.*

HERTZMANN, A., JACOBS, C., OLIVER, N., CURLESS, B., AND SALESIN, D. 2001. Image analogies. In *SIGGRAPH '01*, 327–340.

HERTZMANN, A., OLIVER, N., CURLESS, B., AND SEITZ, S. 2002. Curve analogies. In *Eurographics Workshop on Rendering*, 233–246.

KAJIYA, J. T., AND KAY, T. L. 1989. Rendering fur with three dimensional textures. *Comput. Graph. 23*, 3, 271–280.

KIM, T.-Y., AND NEUMANN, U. 2002. Interactive multiresolution hair modeling and editing. In *SIGGRAPH '02*, ACM, New York, NY, USA, 620–629.

KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2D exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3, 2:1–2:9.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3, 277–286.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. In *SIGGRAPH '05*, ACM, New York, NY, USA, 795–802.

LAGAE, A., DUMONT, O., AND DUTRE, P. 2005. Geometry synthesis by example. In *Proceedings of the International Conference on Shape Modeling and Applications*, 176–185.

LAI, Y.-K., HU, S.-M., GU, D. X., AND MARTIN, R. 2005. Geometric texture synthesis and transfer via geometry images. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, 15–26.

LEFEBVRE, S., AND HOPPE, H. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics 24*, 3, 777–786.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics 25*, 3, 541–548.

LIANG, L., LIU, C., XU, Y., GUO, B., AND SHUM, H.-Y. 2001. Real-time texture synthesis using patch-based sampling. *ACM Trans. Graphics 20*, 3, 127–150.

LLOYD, S. P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2, 129–137.

MARSCHNER, S. R., JENSEN, H. W., CAMMARANO, M., WORLEY, S., AND HANRAHAN, P. 2003. Light scattering from human hair fibers. *ACM Trans. Graph. 22*, 3, 780–791.

PARIS, S., HECTOR M. BRICE N., AND SILLION, F. X. 2004. Capture of hair geometry from multiple images. In *SIGGRAPH '04*, ACM, New York, NY, USA, 712–719.

PARIS, S., CHANG, W., KOZHUSHNYAN, O. I., JAROSZ, W., MATUSIK, W., ZWICKER, M., AND DURAND, F. 2008. Hair photobooth: geometric and photometric acquisition of real hairstyles. In *SIGGRAPH '08*, ACM, New York, NY, USA, 1–9.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *SIGGRAPH '00*, 465–470.

TAKAYAMA, K., OKABE, M., IJIRI, T., AND IGARASHI, T. 2008. Lapped solid textures: filling a model with anisotropic textures. In *SIGGRAPH '08*, ACM, New York, NY, USA, 53.

TURK, G. 2001. Texture synthesis on surfaces. In *SIGGRAPH'01*, 347–354.

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

WATANABE, Y., AND SUENAGA, Y. 1992. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics and Applications 12*, 1, 47–53.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 479–488.

WEI, L.-Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH'01*, 355–360.

WEI, Y., OFEK, E., QUAN, L., AND SHUM, H.-Y. 2005. Modeling hair from multiple views. *ACM Transactions on Graphics 24*, 3, 816–820.

XU, W., ZHOU, K., YU, Y., TAN, Q., PENG, Q., AND GUO, B. 2007. Gradient domain editing of deforming mesh sequences. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, 84.

YU, Y. 2001. Modeling realistic virtual hairstyles. In *Proceedings of Pacific Graphics*, 295–304.

YUKSEL, C., AND KEYSER, J. 2008. Deep opacity maps. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008) 27*, 2.

ZHOU, K., HUANG, X., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2006. Mesh quilting for geometric texture synthesis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006) 25*, 3.

ZHOU, H., SUN, J., TURK, G., AND REHG, J. 2007. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics 13*, 4, 834–848.