A Data-driven Approach to Four-view Image-based Hair Modeling

Meng Zhang Menglei Chai Hongzhi Wu Hao Yang Kun Zhou

State Key Lab of CAD&CG, Zhejiang University*

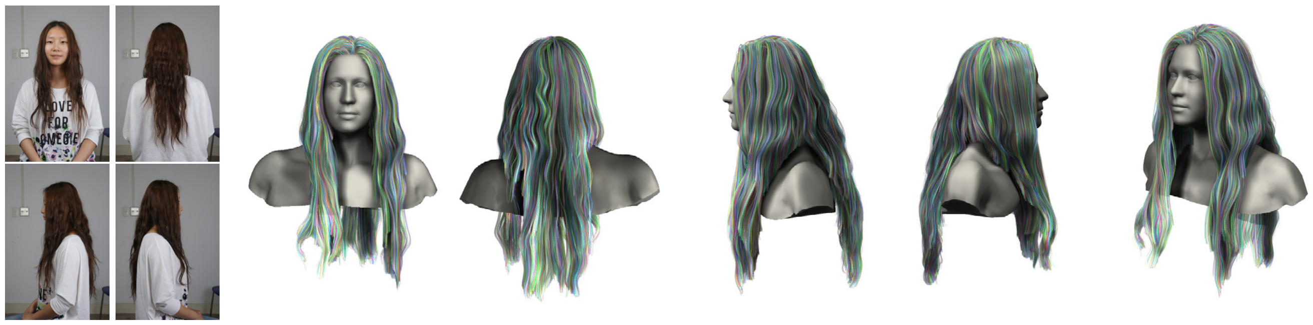


Figure 1: *Given four input hair images taken from the front, back, left and right views, our method computes a complete strand-level 3D hair model that closely resembles the hair in all images.*

Abstract

We introduce a novel four-view image-based hair modeling method. Given four hair images taken from the front, back, left and right views as input, we first estimate the rough 3D shape of the hair observed in the input using a predefined database of 3D hair models, then synthesize a hair texture on the surface of the shape, from which the hair growing direction information is calculated and used to construct a 3D direction field in the hair volume. Finally, we grow hair strands from the scalp, following the direction field, to produce the 3D hair model, which closely resembles the hair in all input images. Our method does not require that all input images are from the same hair, enabling an effective way to create compelling hair models from images of considerably different hairstyles at different views. We demonstrate the efficacy of our method using a wide range of examples.

Keywords: hair modeling, image-based modeling, patch-based texture synthesis

Concepts: •**Computing methodologies** → **Shape modeling;**

1 Introduction

Hair plays a crucial role in producing realistically looking characters in computer-generated imagery. The generation of compelling 3D hairs, however, requires considerable efforts, due to the intricate hair structures and the wide variety of real-world hairstyles. It usually takes a few days of manual work by digital artists to create realistic 3D hair models in the entertainment industry, using dedicated hair design tools.

Recently, significant research effort has been devoted to hair digitalization, to help reduce the laborious work in the hair modeling process. Multi-view hair modeling [Paris et al. 2008; Luo et al. 2013; Hu et al. 2014a] can generate impressive reconstructions of challenging, real-world hairstyles from dozens of images under different views, usually captured in a controlled environment. They often require complex acquisition setups, which are not easily accessible to average users. On the other hand, single-view hair modeling approaches [Chai et al. 2012; Hu et al. 2015; Chai et al. 2016] only take a single hair image as input, and produce plausible 3D hair models by using various forms of priors. As there is no input information about the hair at other views, the modeling result may not match the reality at views distant from the input one. Yu et al. [2014] propose a hybrid image-CAD system to create a 3D hair model from two or three images. Their focus is on visually pleasing results, not close visual matches to the input images.

In this paper, we introduce a novel four-view image-based hair modeling method, to fill in the gap between existing work on multi-view and single-view hair modeling. Given four images taken from the front, back, left and right views as input, our pipeline generates a complete strand-level 3D hair model that closely resembles the hair in all input images, with the help of a small amount of user interaction. We start by estimating the rough 3D shape of the hair observed in the input images, then synthesize a hair texture on the surface of the shape, from which the hair growing direction information on the hair shape surface is calculated and diffused to construct a 3D direction field in the hair volume. Finally, we grow hair strands from the scalp, following the direction field, to produce the 3D hair model.

In designing the above pipeline, we must address one challenging issue – how to generate a 3D hair model that matches *all* four input views with high fidelity. Unlike previous multi-view modeling methods which calculate accurate correspondences between a set of densely captured images, we cannot get reliable correspondences from the sparse set of four images. Simply projecting each image onto the hair shape according to its camera specification and linearly blending in overlapping regions would blur out hair details and might leave holes due to insufficient coverage of the input images. To this end, we propose a novel hair texture synthesis algorithm to effectively combine images from different views to generate a consistent hair texture over the hair surface. Our algorithm is inspired by Image Melding [Darabi et al. 2012], an effective algorithm for combining independent images via patch-based texture synthesis. We generalize the original 2D-image-based algorithm to perform synthesis on a 3D surface from source images with spatial relationships. 3D hair strands are then grown with the help of this hair texture.

One major benefit of our method is that it does not require that all input images are from the same hair, thanks to our multi-source hair texture synthesis algorithm. This is particularly useful when only a

*Corresponding authors: Hongzhi Wu (hongzhi.wu@gmail.com), Kun Zhou (kunzhou@acm.org)

single-view image of the hair is available. We can search over the Internet for plausible and similar hair images at other views, which lowers the requirement for applying our method to create compelling hair models. Furthermore, we could generate interesting hairstyles, by feeding in images of considerably different hairstyles at different views. We demonstrate the efficacy of our method with a wide range of examples.

2 Related Work

Human hair modeling is an extensively studied topic in computer graphics, due to its importance in a variety of applications, as well as its inherent difficulties. Please refer to [Ward et al. 2007] for a comprehensive survey. Here, we limit our discussion to a small number of representative methods mostly related to our work.

Geometry-based Hair Modeling. Related papers (e.g., [Kim and Neumann 2002; Yuksel et al. 2009]) introduce design tools to directly create and edit the hair geometry. Wang et al. [2009] propose a hair geometry synthesis approach to hair modeling. Given an input 3D hair model, they can create a novel one with a statistically similar spatial arrangement of hair strands and geometric details. A key component of our method is a hair texture synthesis algorithm that generates a consistent hair texture over the surface of the hair shape from four input images.

Multi-view Hair Modeling. This category of methods creates high-quality 3D hair models from images taken from a large number of different views. They often require complex acquisition setups, such as a number of cameras as well as related controllers. Wei et al. [2005] take as input about 40 images of the same hair, and calculate correspondences among all input images to estimate hair orientations. Paris et al. [2004] recover 3D fiber orientations for hair reconstruction by analyzing image sequences with difference lighting conditions, taken from four views. Paris et al. [2008] propose an active hair capturing system, which can acquire the positions of exterior hair strands with high precisions. Jakob et al. [2009] capture detailed arrangement of fibers in a hair assembly by growing the strands within the diffused orientation field from the scalp to the exterior hair layer. In [Luo et al. 2013], coherent and plausible structure-aware wisps are reconstructed from multi-view images, which are used to robustly synthesize hair strands. Herrera et al. [2012] use thermal imaging to generate strands by growing from the boundary of the captured hair. Recently, Hu et al. [2014a] propose a strand fitting algorithm to find structurally plausible configurations among the locally grown hair segments using a database of simulated examples. Cao et al. [2016] constructs a coarse geometric proxy as a morphable hair model from 32 images for a user. All these multi-view modeling methods require densely captured images to calculate dense correspondences between images and solve for the hair geometry. We tackle the unique challenge of unreliable correspondences among a sparse number of views using a hair texture synthesis approach.

Yu et al. [2014] propose a hybrid image-CAD system to create and edit hair models from two or three images. In a concurrent work, Vanakittistien et al. [2016] introduce a lightweight system, which takes as input 8 photos and generates a 3D hair model via guide strand tracing on a 3D orientation field computed from multi-view photos using [Wei et al. 2005]. Here one key difference is that these methods model 3D hair that is visually pleasing but not necessarily faithful to input in details, while our results visually resemble all input images captured at different views. Moreover, in our system, an average user can easily create novel hairstyles by combining different hair images from different subjects. It is not known how to efficiently perform the same task in [Yu et al. 2014; Vanakittistien et al. 2016].

Single-view Hair Modeling. Considerable progress has been made to the development of single-image-based hair modeling techniques. Chai et al. [2012] introduce an effective high-precision 2D strand tracing algorithm, which is used to further create 3D strands that match inter-strand occlusions. The system is extended to handle single video input in [Chai et al. 2013]. Hu et al. [2015] first retrieve a 3D model from a large database, and then perform joint optimization on strands taking into account 2D similarity, physical plausibility and orientation coherence. Chai et al. [2015] exploit shading cues in the input image to produce high-quality results. More recently, a fully automatic single-view hair modeling pipeline is proposed in [Chai et al. 2016], which uses deep neural networks to estimate the hair region and hair growing directions. However, it is not clear how to extend their paper to our multi-view case. Due to the potentially complex interactions between hair and body (e.g., Fig. 1), it is much more difficult to find a good detailed hair model that matches all four views in a limited database, using the mask-based search in [Chai et al. 2016]. It is even more difficult to combine physically inconsistent hairstyles at different views.

One problem with single-view hair modeling techniques is the lack of control over the final result at views distant from the input one (e.g., the back view), as there is no input information at all. In comparison, our method lets the user explicitly provide additional hair images at the side and back views, which enables full control over the final hair model at different views.

PatchMatch-based Texture Synthesis. Our hair texture synthesis algorithm is motivated by the PatchMatch algorithm [Barnes et al. 2009; Barnes et al. 2010], which efficiently computes a Nearest-Neighbor Field (NNF) that stores the correspondences between patches of a 2D image, as well as the Image Melding technique that builds on PatchMatch to combine inconsistent 2D images [Darabi et al. 2012]. We adapt the original PatchMatch and Image Melding algorithms to our specific domain, where the input is four images and their spatial relationships are described by corresponding camera specifications.

3 Overview

Given four hair images taken from the front, back, left and right views as input, our method first constructs a rough 3D hair shape: we segment the regions that contain the hair in all input images, and roughly specify the view condition for each image; then we search in a predefined 3D hair database and select a model that best matches the hair contours at different views; the model is further deformed to better align with the hair contours and clipped against the hair-face boundaries. Next, we synthesize a consistent hair texture over the surface of the hair shape, by fusing information from different hair images. We subsequently compute hair directions on the hair shape surface and propagate them into the entire volume, which results in a 3D direction field. Guided by this direction field, we trace and generate 3D hair strands from the scalp, and refine the details by piecewise helix fitting. The output of our method is a consistent, detailed strand-level 3D hair model, which closely resembles the hair in the input images.

4 Our Hair Modeling Pipeline

We describe our hair modeling pipeline in details. Please refer to Tab. 1 in the appendix for a list of notations used in this section.

Hair Segmentation. Given four input images $\{I_i\}, (i = 1, 2, 3, 4)$ that depict the hair from the front, back and two side views, the first step in our method is to segment the regions that contain the hair. We adopt the user-assisted tool in [Li et al. 2004] to efficiently perform this task. Other popular segmentation tools can also be

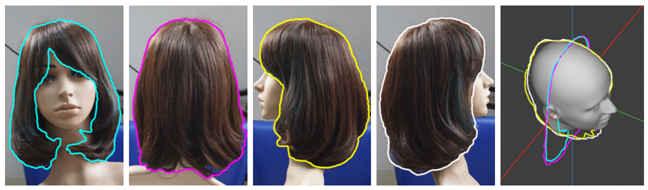


Figure 2: *Camera specification. For the four images shown on the left (with hair contours marked), the user roughly adjusts their sizes and orientations in a 3D coordinate system with a reference head model. Our UI is shown on the right. Camera parameters T_i for each image can be computed once the adjustment is finished.*

used here. The resultant binary hair mask is denoted as $\{M_i^I\}$. We set the value of a pixel of the mask to 1 if it is in the regions that contain the hair and 0 otherwise.

Camera Specification. The next step is to roughly specify the camera configuration for each image, assuming that the camera is orthographic. We develop a simple user interface to facilitate this task. Please see Fig. 2 for an illustration. For each segmented hair image, the user first roughly selects a view direction (i.e., front, back, left-side or right-side); then the hair contour of the image, obtained from the previous segmentation step, is projected onto a 3D plane, which orients toward the user-specified view direction that passes through the center of a reference 3D head model. The transformation of this reference head model is computed with [Blanz and Vetter 1999] for the subject in the front-view image to match the facial feature points. With all contours from different views visualized along with the reference head model, the user can further fine-tune the view direction, or translate, rotate or scale each hair contour to make it consistent with others. The corresponding camera parameters T_i for each image I_i can be directly computed, once the user finishes adjusting the hair contour in the 3D space. Note that our modeling algorithm does not require accurate specifications of the camera parameters, which will be demonstrated in Sec. 5.

4.1 Rough Hair Shape Construction

Once we obtain estimations of camera configurations for all four input images, we use this information along with the hair contours to build a rough 3D shape that represents the coarse-scale hair geometry. The detailed hair strands are generated in later stages of our pipeline, where issues like potential inconsistencies in the hair among different input images are resolved. Note that constructing a 3D shape solely from as few as four images is highly challenging. Therefore, similar to previous work [Hu et al. 2015; Chai et al. 2016], we use a data-driven approach to build the shape with the help of a predefined database of 3D hair models, detailed as follows. An example of the construction process is shown in Fig. 3.

Database Model Search. We search in the 3D hair database of [Chai et al. 2016] for the model G_{DB} that minimizes the sum of distances between the input hair masks M_i^I and the hair masks in the database M_i^H, based on the distance field D_i^I of the hair mask M_i^I as:

$$dist(\{M_i^I\},\{M_i^H\}) = \sum_i \frac{\sum_{j\in M_i^I \oplus M_i^H} |D_i^I(j)|}{A(M_i^I)}, \tag{1}$$

where i denotes the view, M_i^H is a binary mask generated by rendering a database model using the camera parameters T_i, and $A(\cdot)$ computes the area of a mask. $M_i^I \oplus M_i^H$ represents the symmetric difference between two masks. Please refer to the supplemental material for a visualization of the database.

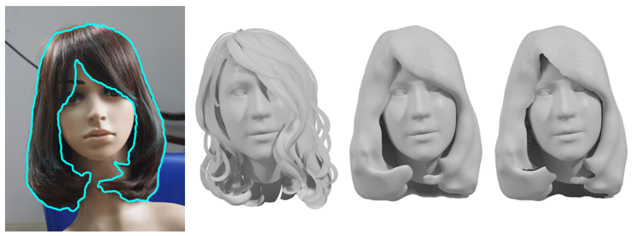


Figure 3: *Rough hair shape construction. From left to right, the front-view image with hair contour marked, the model retrieved from the database, the result after detail removal and mask-based deformation, and the final hair shape after visibility- and boundary-based clipping.*

High-frequency Detail Removal. Once the best database model G_{DB} is retrieved, we need to remove its excessive high-frequency geometric details, as typically they are not consistent with the hair in the input images, even though the rendered masks of the model $\{M_i^H\}$ are close to the hair masks $\{M_i^I\}$ at different views. To do so, we first convert G_{DB} into a Signed Distance Field (SDF) with a resolution of $100\times100\times100$ over the bounding cube of the model [Zhao 2005]: after the voxelization step of the conversion, we blur the details by filtering the voxels (0 for empty voxels and 1 for non-empty ones) with a 3D Gaussian kernel ($\sigma = \frac{3}{100}b$, where b is the length of the bounding cube of the model), and perform thresholding to transform the result back to a binary form. Finally, we apply the marching cube algorithm [Lorensen and Cline 1987] to obtain a closed triangular mesh G_{MC} from the SDF.

Mask-based Deformation. Due to the complexity of real-world hairstyles, it is inevitable that the database model we retrieve does not align perfectly with the hair contours in segmented images. Moreover, the high-frequency detail removal step introduces additional changes to the shape. To alleviate this issue, we deform our current mesh G_{MC} to match the hair masks from different views, by extending the formulation in [Chai et al. 2016] to the multi-view case. Specifically, we deform each vertex p in G_{MC} to p' by minimizing the following energy:

$$\begin{aligned}\arg\min_{p'} \sum_{p_j\in G_{MC}} [\sum_{i=1,2,3,4} ||P_{d_i}(p_j{}' - W_i(p_j))||^2 \\ + \lambda||\Delta p_j{}' - \frac{\delta_j}{|\Delta p_j{}'|}\Delta p_j{}'||^2],\end{aligned} \tag{2}$$

where P_d projects a vector along a direction d:

$$P_d(p) = p - (p\cdot d)d.$$

Here the first term in Eqn. (2) measures the deviation of a vertex to its deformation target along each view direction d_i, and the second is a regularization term to preserve local geometric features in the original shape. W_i is a warping function computed with thin plate spline, Δ is the discrete mesh Laplacian operator based on the cotangent formula, and δ_j is the magnitude of Laplacian coordinates of vertex v_j in G_{MC}. We set $\lambda = 1$ in all experiments. The result after the deformation is denoted as G_{DEF}.

Visibility- and Boundary-based Clipping. We first remove the inner surface of the rough hair shape G_{DEF}: we connect a line segment from the center of the head to each vertex; if this line segment intersects with another part of G, we keep the vertex as it belongs to the outer surface of G; otherwise we remove the vertex. Furthermore, we clip the result against the hair-face boundaries in $\{I_i\}$ for each view. This step further improves the geometric quality, resulting in the final 3D rough hair shape G. An example can be found in Fig. 3.

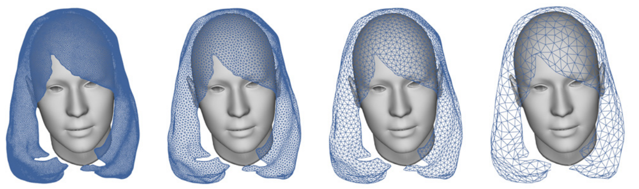


Figure 4: *Remeshing results by uniformly sampling new vertices with different densities, which will later be used in hair texture synthesis.*

4.2 Hair Texture Synthesis from Multiple Sources

After the rough hair shape G is obtained, we synthesize a consistent hair texture over its surface using the input images as sources. This texture will be used to construct a 3D direction field that guides the hair strand generation. The simplest way to obtain a hair texture is to project each I_i onto the hair shape based on its camera T_i and linearly blend in overlapping regions. But the quality of the result would be limited: due to unreliable correspondences among input images, direct linear blending is likely to blur intricate hair details, which leads to inaccuracies in subsequent hair direction estimation; there might be holes due to insufficient coverage of input images. To produce a high-quality hair texture from multiple sources, we propose a patch-based hair texture synthesis approach inspired by Image Melding [Darabi et al. 2012].

Directly applying the image melding technique to our problem is not feasible, due to two key differences. First, the original technique generates a regular 2D image as output, while our resultant texture is defined on a non-regular 3D surface. Second, the sources in [Darabi et al. 2012] are independent 2D images, while each of our source images is associated with a camera that describes its spatial relationship with other images. We thus develop a texture synthesis algorithm that takes into account these differences, described as follows.

Preprocessing. Specifically, we first perform remeshing by uniformly sampling new vertices $\{p_j\}$ over the surface of G via the method in [Valette and Chassery 2004]. An example is shown in Fig. 4. Next, we compute a uv-mapping of G using the UVAtlas tool [Microsoft 2017], to parameterize the hair texture. For each p_j, we define its associated patch of the size $m \times m$, by projecting a region of the texture around p_j along the normal direction to its tangent plane. A patch can be viewed as a local resampling of the hair texture, following previous work such as [Wei and Levoy 2001]. Each pixel on a patch corresponds to a 3D point on G, which subsequently corresponds to a texel based on the uv-parameterization. Note that multi-resolution versions of vertices and the texture are created for later processing. We use four scales in all experiments. The hair texture I is initialized by projecting each I_i on to G and then update corresponding texels; the closest view with respect to the surface normal is selected, if a texel corresponds to multiple projections; for holes not covered by the projections of input images, we fill them using interpolations from hole boundaries with inverse distance weighting [Wexler et al. 2007].

Main algorithm. Similar to [Darabi et al. 2012], our hair texture synthesis algorithm repeats the following process for a user-specified number of iterations at each scale in a coarse-to-fine fashion. First, we perform single-source patch-based synthesis (denoted as SingleSourceSynthesis in Algorithm 1 and described in Sec. 4.3) separately for each input image I_i, to fill in the regions of surface on G not covered by the projection of I_i. Second, the colors and gradients from the synthesis results of each input image are blended based on view-dependent weights α_i, computed as the dot product between a vertex normal corresponding to a texel and the inverse view direction, and further normalized to ensure that $\sum_i \alpha_i = 1$. Finally, we update the hair texture colors based on the gradients by solving the screened Poisson equation [Bhat et al. 2008]. Please see Algorithm 1 for details. An example of the synthesis result is shown in Fig. 5.

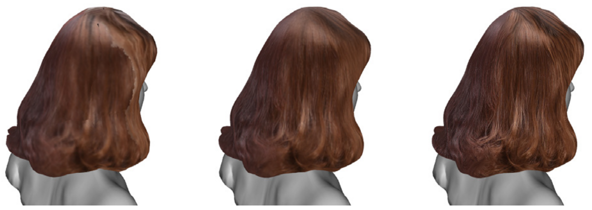


Figure 5: *Progress of our hair texture synthesis (cf. the second row of Fig. 14 for input images). The left image is the texture after the initialization, the center image is the intermediate result after half of the total iterations, and the right image shows the final synthesis result. Our synthesis algorithm resolves the inconsistencies in the initial texture due to multi-view sources, and produces a high-quality final result.*

ALGORITHM 1: Hair texture synthesis algorithm

```
for scale = 1 → 4 do
    for iteration = 1 → n do
        for i = 1 → 4 do
            $\overline{I}_i \leftarrow$ SingleSourceSynthesis($I$, $I_i$)
        end for
        $\overline{I} = \alpha_1\overline{I}_1 + \alpha_2\overline{I}_2 + \alpha_3\overline{I}_3 + \alpha_4\overline{I}_4$
        $j \leftarrow \arg\max \alpha_j|\nabla\overline{I}_j|$
        $\overline{\nabla I} \leftarrow \overline{\nabla I}_j$
        $I \leftarrow$ ScreenedPoisson($\overline{I}$, $\overline{\nabla I}$)
    end for
end for
```

4.3 Single-source Patch-based Synthesis

Now we describe in details the single-source patch-based synthesis for each input image I_i, denoted as SingleSourceSynthesis(R,S), which is the key part in hair texture synthesis from multiple sources. Similar to [Darabi et al. 2012], our goal is to fill the contents in a target region R with those from the source S. To do so, we minimize the sum of squared differences between corresponding texels via patch-based optimization:

$$E(R,S) = \sum_{U(C(X))\in R} \min_{U(C(Y))\in S} ||Y - X||^2. \qquad (3)$$

Here we divide the resultant texture into S and R. We determine if a texel belongs to S by testing the following two conditions: (1) the corresponding 3D point on G can be back-projected with no obstacles to M_i^I under the camera T_i; (2) the angle between the corresponding normal and the direction of back-projection is below a threshold θ ($\theta = \frac{\pi}{4}$ in our experiments). $C(\cdot)$ returns the center point p corresponding to a patch, $U(\cdot)$ returns the texel corresponding to a 3D point based on the uv-parameterization on G. Y is an $m \times m$ patch, and $X = f(N(p))$ is an $m \times m$ patch after applying a geometric transformation f on a small neighborhood N around a source point p. We define f to be the concatenation of an in-tangent-plane scaling in the range of [0.9, 1.1], and an in-tangent-plane rotation in the range of $[-\frac{\pi}{15}, \frac{\pi}{15}]$.

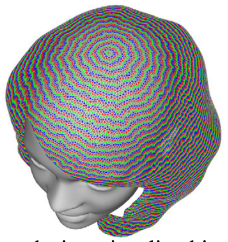
To solve this optimization, we alternate between two steps, patch search and color voting, as in [Darabi et al. 2012]. We first define a source patch X as a patch that satisfies $\forall x \in X, U(x) \in S$. Any patch that does not meet this condition is classified as a target patch. In the patch search step, we find the most similar source patch for every target patch. Next, in the color voting step, each patch casts votes on every texel, which are then weighted averaged to generate a new texture.

4.3.1 Patch Search

The first step in the optimization is to search for the source patch that best matches a given target patch. Generalized PatchMatch [Barnes et al. 2010] is adopted in [Darabi et al. 2012] to perform this task efficiently. However, PatchMatch works on regular 2D images and cannot be extended to our problem in a straightforward manner. Therefore, we develop an approach that changes the following key components in the original algorithm, in order to adapt to our domain.

NNF. Similar to PatchMatch, we compute an NNF over all vertices. At each vertex p_j, the NNF stores a 2D coordinate on the input image I_i, denoted as g_j.

Initialization. If $U(p_i) \in S$, then we can directly obtain g_j by back-projecting p_i to the image plane. Otherwise, the offsets are initialized with random coordinates inside the hair mask.



Scan Order. We no longer have a native scan order for applying PatchMatch over vertices on a 2-manifold, while the original work processes in the scan-line order. To solve this issue, we start with a vertex on the top of the rough hair shape of hair, and then travel its n-th ring neighborhood in the counter-clockwise order (n = 1, 2, ...), until all vertices are processed. An example is shown in the inset figure, with each ring visualized in a same color.

Propagation. The NNF at a vertex p_j could get improved by propagating the information from its one-ring neighborhood. However, it is not trivial to apply the original 2D image-based propagation: the offset between two 3D vertices (p_j and its spatial neighbor) is not of the same measure as the offset on the input 2D image. To tackle this problem, we harness the additional information of the camera T_i associated with I_i and convert the 3D offset into a 2D one via a transformation. Please see Fig. 6 for a graphical illustration. Specifically, we suppose that the patch match error at a vertex p_k is smaller than that at p_j. To update the NNF g_j with g_k, we first project the pixel on I_i corresponding to g_k to G using T_i, and denote the projected 3D point on G as $q = T_i g_k$. Next, we compute an affine transformation $\hat{T}_k$ (translation and rotation), which transforms the local frame at p_k to that at q. Then we can update g_j as $g_j = T_i^{-1}\hat{T}_k p_j$. Note that in analogy to the original work, we check the visited / unvisited one-ring neighborhood in counter-clockwise order at odd / even iterations. The tangent at each vertex is computed as the normalized cross product of the normal and the positive y direction. This simple method works well in all our experiments. More advanced techniques such as [Fisher et al. 2007] can also be employed here.

Random Search. A similar problem arises in directly applying the original random search to our domain, due to the mismatch in 3D and 2D offsets. We also tackle this issue with the help of T_i. Specifically, we try to improve the NNF g_j corresponding to the vertex p_j, by testing the differences between the current patch X_j and a series of candidate patches $\{Z_k\}$. To construct a Z_k, we first compute a random offset u_k as:

$$u_k = g_j + w\beta^k r_k, \tag{4}$$

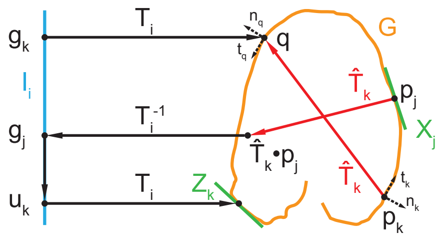


Figure 6: *An illustration of propagation and random search. To propagate the offset g_k of p_k to p_j, we first compute $\hat{T}_k$ that transforms the local frame at p_k to q. Then we apply the same transform to p_j and back-project to the image plane, resulting in the updated offset g_j. To perform random search, we first compute a random offset u_k, and then obtain its associated patch Z_k by projecting from the image plane onto the rough hair shape G. Next, Z_k is compared with X_j to check if it can improve the NNF at p_j.*

where r_k is a 2D uniform random variable in $[-1, 1]^2$, w is a maximum search distance (w = the dimension of I_i), and $\beta = 0.5$ is a fixed ratio between search window sizes. The process is repeated until $w\beta^k$ is below one pixel. Once u_k is computed, we project it onto G using T_i, and denote the patch centered at the intersection as Z_k. Please see Fig. 6 for a graphical illustration.

4.3.2 Color Voting

Once the patch search is finished, the second step in our single-source synthesis is color voting. Similar to [Darabi et al. 2012], the optimal texture $\overline{I}_i$ is computed as the weighted average of the corresponding pixels in all overlapping patches:

$$\overline{I}_i(t) = \frac{\sum_j \gamma_j X_j(p(t))}{\sum_j \gamma_j}. \tag{5}$$

Here t represent 2D texel coordinates, and $p(t)$ is the 3D point on G that corresponds to the texel $\overline{I}_i(t)$. $\{X_j\}$ is the set of all patches, whose projection onto G contains $p(t)$. We denote the patch texel on X_j, whose projection is $p(t)$, as $X_j(p(t))$. γ_j is a view-dependent weight, computed as the dot product between the normal at $p(t)$ and the normal of the patch X_j; the result is further clamped to zero, if it is less than 0.5.

4.4 Hair Direction Field Construction

After we obtain a hair texture over the surface of the hair shape G, we compute a 3D hair direction field to guide hair strand generation in the subsequent stage, similar to existing work such as [Chai et al. 2013].

First, we apply the iterative refinement method in [Chai et al. 2012] on each texel of the hair texture to obtain a hair direction texture. Note that here we work on the tangent plane at the 3D point corresponding to a texel, rather than in the texture space directly. We then render the result at five different views: front, back, left-side, right-side and top. Directional ambiguities due to issues like imperfection of the imaging process are resolved by user strokes on individual views, which roughly indicate the hair growth directions, similar to [Chai et al. 2013]: the user strokes are projected onto the hair direction texture, based on G; if there are more than one user stroke over a texel, we select one from the closest view with respect to the normal corresponding to the texel. With user strokes as directional constraints, we update the hair direction texture via binary integer programming, as in [Chai et al. 2013]. Finally, we construct a uniform 3D direction field inside the volume of the closed mesh

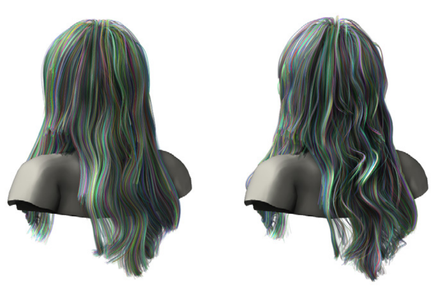


Figure 7: *Detail refinement. The left image shows a model without detail refinement, and the right one is generated with the refinement, exhibiting richer details.*

G_{DEF}, using the hair direction texture as constraints on the surface of the rough hair shape and minimizing a least-squares energy function as in [Chai et al. 2013].

4.5 Hair Strand Generation

With the help of the hair direction field, we generate 3D hair strands using a two-step method. In the first step, similar to [Chai et al. 2013], we uniformly sample a default scalp region on the reference head model to grow 2,000$\sim$2,500 guide strands, according to the directions stored in the hair direction field. The growing is terminated when a maximum curve length is exceeded or the strand is outside of G_{DEF}. The second step adds new strands that are as visually and physically plausible as the guide strands, in order to fill in the remaining empty space in the volume. For each new strand, we again uniformly sample the scalp region as the starting point, then copy the nearest guide strand. After that, we adopt the linear blend skinning approach in [Hu et al. 2015] to deform the new strand to be coherent with neighboring ones. Note that we enforce the mask constraints in strand generation, so that the projection of each strand using T_i must stay inside the hair mask M_i^I for i = 1, 2, 3 and 4. We use a fixed strand number of 30,000 in our experiments.

4.6 Detail Refinement

As our hair direction texture is defined over the surface of G, the hair direction at each point lies within the corresponding tangent plane; there is no information about the component of the hair direction along the normal direction. We alleviate this problem by extending the detail refinement technique in [Hu et al. 2015] from a single image domain to a 2-manifold, which fits piecewise helix curves to the guide strands. Please see Fig. 7 for an example.

Specifically, we first propagate the normals from the surface of G to the entire hair shape volume, using the same method as we construct the hair direction volume from the hair texture [Chai et al. 2013], which results in a normal volume that indicates the directions of refinement. Next, for each guide strand, we compute the normal at each of its vertex by interpolating from the normal volume, and divide the strand into a few segments based on normal variations: for each segment, the angle between the normal at each vertex and the average normal is below a threshold (we use 25° in all experiments). Finally, we project each strand segment to a plane perpendicular to the average normal, and fit a piecewise helix curve with the method of [Cherin et al. 2014].

5 Experimental Results

All experiments are conducted on a workstation with an Intel i5-4590 CPU and 32GB of memory. For a typical input set of four

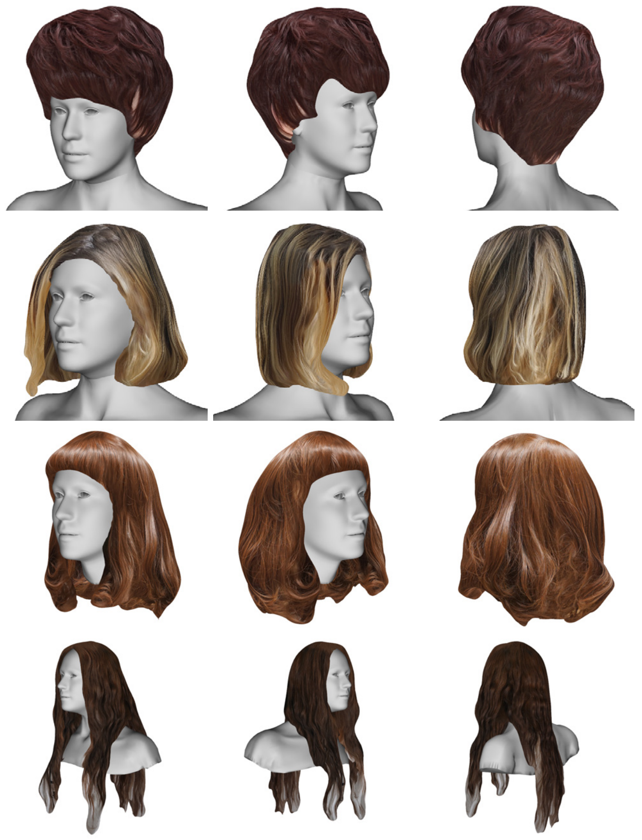


Figure 8: *Hair texture synthesis results from different views. Input images are from the first row of Fig. 14, the last row of Fig. 17, Fig. 11 and Fig. 1, respectively.*

800$\times$800 images, the total processing time using our unoptimized pipeline is about 25 minutes. The majority of time spends on the patch-based synthesis of a hair texture of 2048$\times$2048, which takes 14 minutes. We use 20, 10, 5 and 2 iterations for 4 scales in hair texture synthesis, respectively. It takes about 6 minutes in total to finish all user interactions in our pipeline, including hair segmentation, camera specification and hair direction disambiguation. We show the user input for each example in this paper, as well as a video on a complete interactive editing session, in the supplemental materials.

Modeling. We first demonstrate in Fig. 1 and 17 the effectiveness and generality of our method with the modeling results on a variety of hairstyles, ranging from short / straight to long / curly. Each result is computed using four-view images of the same hair. As shown in Fig. 1 and 17, the 3D strand-level hair models produced by our method closely resemble the hair at all input views, and the resultant hair strands are spatially consistent. Please also refer to the accompanying video for more details.

Thanks to our multi-source patch-based synthesis algorithm, we do not require that the input images are from exactly the same hair, as the slight inconsistencies among the input images can be resolved in the hair texture synthesis. This enables a novel application for hair modeling using Internet images: given a single-view hair image, the user can manually search over the Internet for images taken from other views, which look similar to the hair in the first image. Our approach produces plausible 3D results in this case, as shown in Fig. 14. Furthermore, we can even "meld" highly different hairs at different views, to create novel, interesting hairstyles, such as

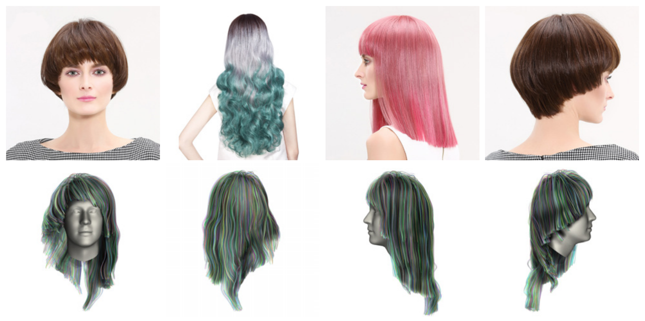


Figure 9: *A failure case due to multi-view inconsistency. Input images are shown in the top row, and the modeling result in the bottom. The main reason for the failure is that physically there is no rough hair shape that can match the hair masks in all views.*

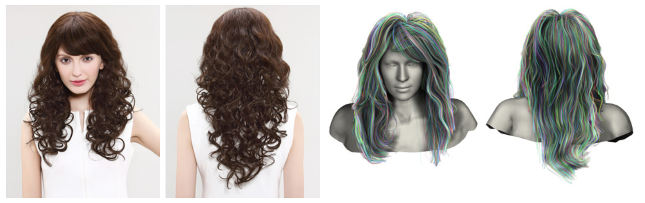


Figure 10: *A failure case on a highly curly hairstyle. From left to right, the front / back input images, and the modeling results at corresponding views.*

one that is curly at the front view and straight at the back, shown in Fig. 14. Again the considerable inconsistencies are well resolved, using our hair texture synthesis algorithm. Please also refer to the accompanying video for demonstrations.

Although our pipeline can combine different hairs, we require that the hair masks at four input images roughly correspond to a plausible 3D shape. Otherwise, it is physically impossible to satisfy all the mask constraints, which leads to poor results shown in Fig. 9. In addition, highly curly hairstyles are known to be challenging for single-image-based hair modeling. Our approach cannot produce high-quality results for such hairstyles. A failure case is shown in Fig. 10.

Comparisons. We first compare our method with AutoHair [Chai et al. 2016], a state-of-the-art single-view-based hair modeling technique, in Fig. 15. AutoHair relies on the retrieved model from a 3D hair model database to generate hair strands at the back or side views, where there is no input information. In comparison, our approach explicitly takes as input four images from different views, and produces a hair model that closely resembles the hair at all views. Please refer to the accompanying video for more detailed comparisons.

In Fig. 16, we compare our method with a state-of-the-art multi-view hair modeling method [Hu et al. 2014a]. The high-quality result using their method is computed with 66 images, simultaneously captured with a setup consisting of the same number of DSLRs. In comparison, we use only 4 of the original images as input to our algorithm. The result looks plausible and closely matches the input images. Note that our method does not require any complex setup for image capturing.

In addition, we compare with the lightweight modeling technique from [Vanakittistien et al. 2016] using four more photos as input. Please see Fig. 11 for a detailed comparison. Note that even for views used as input in [Vanakittistien et al. 2016] but not in our approach, our result exhibits considerably higher quality with realistic

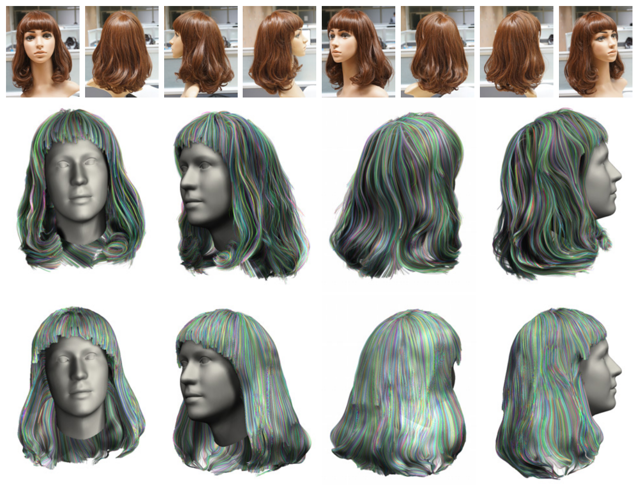


Figure 11: *Comparisons with [Vanakittistien et al. 2016]. From top to bottom, input images, our result using the first 4 input photos in the first row, and the result of [Vanakittistien et al. 2016] using all 8 photos. Our result shows more realistic details even for views not used in our input (i.e., the second / third column).*

details close to those in the photos.

Evaluation. To evaluate the efficacy of our hair texture synthesis algorithm, we show in Fig. 8 the synthesis results on a variety of cases, including those with different hairs at different views. The results are consistent and of high quality, which makes it possible to generate compelling 3D hair models by applying subsequent processing steps.

To evaluate the robustness of our approach with respect to errors in camera specifications of input images, we add unbiased Gaussian noise to the view directions in one controlled experiment, where the ground-truth camera parameters are calibrated. Specifically, we perturb the view direction of each image with an angle sampled from a Gaussian distribution with a standard deviation of σ. As can be seen in Fig. 12, our approach is robust with respect to the errors in view directions. The result is still of visually high quality even when $\sigma = 25^\circ$.

We also evaluate the impact of the number of input images in Fig. 13. Thanks to the novel, robust hair texture synthesis algorithm, our system still produces plausible results with as few as two input images. Adding more input images results in more complete control over the final hair model, as shown in the figure.

6 Conclusions and Future Work

We have presented a lightweight, image-based hair modeling method that takes as input only four hair images at the front, back, left and right views, and produces a high-quality strand-level 3D hair model. Our result closely resembles the hair at all input views. The core of our pipeline is a novel patch-based multi-source hair texture synthesis algorithm, which enables creative hairstyle design by combining different hairstyles at different views into a consistent 3D hair model.

Our work is subject to a number of limitations, which may inspire interesting future work. First, it would be desirable to completely automate our pipeline, by extending the work of [Chai et al. 2016] with additional training data on side- and back-view hair images, as well as associated camera parameters. Moreover, it would be interesting to take into consideration the *appearance* of hair at different

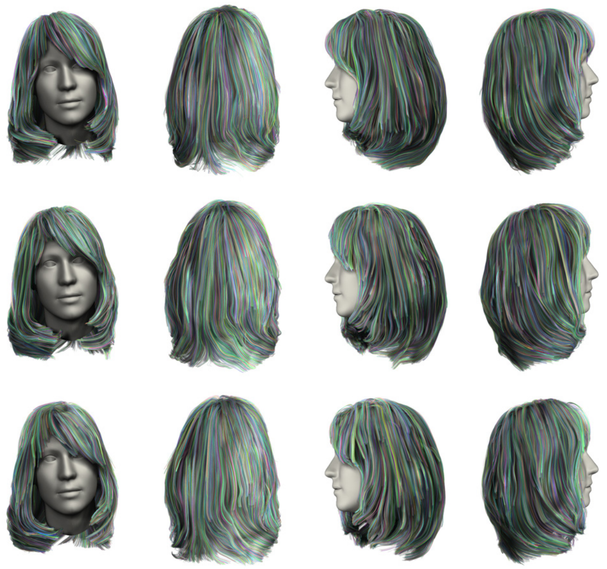


Figure 12: *Impact of camera specification errors. Gaussian noise with $\sigma = 0^\circ/10^\circ/25^\circ$ (the first/second/third row) is added to the ground-truth view direction of each input image (cf. Fig. 2). The corresponding modeling results are shown in each row.*

views, in addition to geometric information, so that cool effects like multi-color-dyed hairstyles can be created. Finally, unlike [Hu et al. 2014b], we do not handle constrained hairstyles such as braids and buns. It would be useful to extend our approach to process these complicated cases.

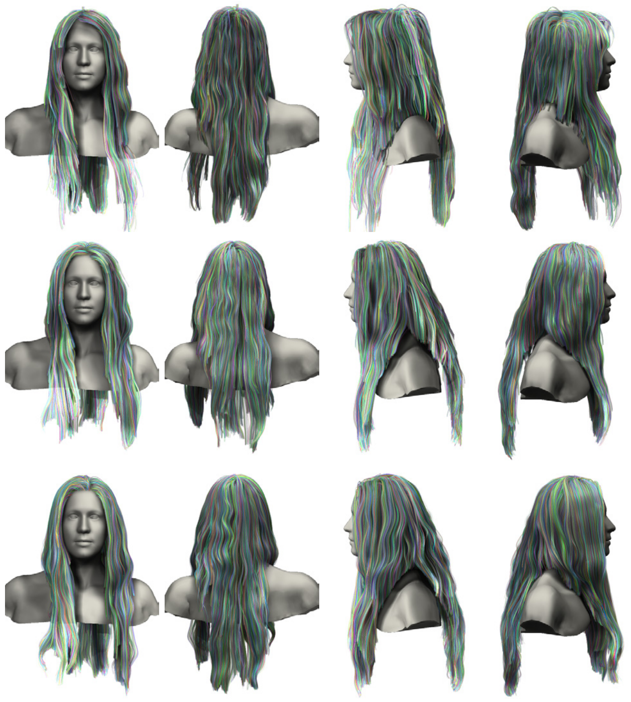


Figure 13: *Impact of the number of input images. From top to bottom, our modeling results with 2 (front/back), 3 (front/back/left) and 4 (front/back/left/right) input images, respectively (cf. input images in Fig. 1).*

Acknowledgements

We would like to thank Qi Zhang, Cassandre Bourdon, Linli Huang, Laura Schmidt, Thauana de Morais, Nanna Carstens and Angelique Wu for being our hair models, Yanliang Li from RebeccaFashion for generously providing hair images, Liwen Hu and Hao Li for help with comparisons, the artists for making their hair models available on The Sims Resource and Newsea platform, and anonymous reviewers for constructive comments. This work is supported by the National Science Foundation China (U1609215), the National Program for Special Support of Eminent Professionals of China, and the Fundamental Research Funds for the Central Universities (2017XZZX009-03).

References

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. 28*, 3 (July), 24:1–24:11.

BARNES, C., SHECHTMAN, E., GOLDMAN, D. B., AND FINKELSTEIN, A. 2010. The generalized patchmatch correspondence algorithm. In *ECCV'10*, 29–43.

BHAT, P., CURLESS, B., COHEN, M., AND ZITNICK, C. L. 2008. ECCV. 114–128.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3D faces. In *Proc. SIGGRAPH '99*, 187–194.

CAO, C., WU, H., WENG, Y., SHAO, T., AND ZHOU, K. 2016. Real-time facial animation with image-based dynamic avatars. *ACM Trans. Graph. 35*, 4, 126.

CHAI, M., WANG, L., WENG, Y., YU, Y., GUO, B., AND ZHOU, K. 2012. Single-view hair modeling for portrait manipulation. *ACM Trans. Graph. 31*, 4, 116.

CHAI, M., WANG, L., WENG, Y., JIN, X., AND ZHOU, K. 2013. Dynamic hair manipulation in images and videos. *ACM Trans. Graph. 32*, 4, 75.

CHAI, M., LUO, L., SUNKAVALLI, K., CARR, N., HADAP, S., AND ZHOU, K. 2015. High-quality hair modeling from a single portrait photo. *ACM Trans. Graph. 34*, 6, 204.

CHAI, M., SHAO, T., WU, H., WENG, Y., AND ZHOU, K. 2016. Autohair: Fully automatic hair modeling from a single image. *ACM Trans. Graph. 35*, 4, 116.

CHERIN, N., CORDIER, F., AND MELKEMI, M. 2014. Modeling piecewise helix curves from 2d sketches. *Computer-Aided Design 46*, 258 – 262.

DARABI, S., SHECHTMAN, E., BARNES, C., GOLDMAN, D. B., AND SEN, P. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph. 31*, 4 (July), 82:1–82:10.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Trans. Graph. 26*, 3 (July).

HERRERA, T. L., ZINKE, A., AND WEBER, A. 2012. Lighting hair from the inside: A thermal approach to hair reconstruction. *ACM Trans. Graph. 31*, 6, 146.

HU, L., MA, C., LUO, L., AND LI, H. 2014. Robust hair capture using simulated examples. *ACM Trans. Graph. 33*, 4, 126.

HU, L., MA, C., LUO, L., WEI, L.-Y., AND LI, H. 2014. Capturing braided hairstyles. *ACM Trans. Graph. 33*, 6 (Nov.), 225:1–225:9.

HU, L., MA, C., LUO, L., AND LI, H. 2015. Single-view hair modeling using a hairstyle database. *ACM Trans. Graph. 34*, 4, 125.

JAKOB, W., MOON, J. T., AND MARSCHNER, S. 2009. Capturing hair assemblies fiber by fiber. *ACM Trans. Graph. 28*, 5, 164.

KIM, T.-Y., AND NEUMANN, U. 2002. Interactive multiresolution hair modeling and editing. *ACM Trans. Graph. 21*, 3 (July), 620–629.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *ACM Trans. Graph. 23*, 3 (Aug.), 303–308.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87*, ACM, SIGGRAPH '87, 163–169.

LUO, L., LI, H., AND RUSINKIEWICZ, S. 2013. Structure-aware hair capture. *ACM Trans. Graph. 32*, 4, 76.

MICROSOFT, 2017. Uvatlas. https://github.com/Microsoft/UVAtlas/.

PARIS, S., BRICEÑO, H. M., AND SILLION, F. X. 2004. Capture of hair geometry from multiple images. *ACM Trans. Graph. 23*, 3 (Aug.), 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. *ACM Trans. Graph. 27*, 3, 30.

VALETTE, S., AND CHASSERY, J.-M. 2004. Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. *Computer Graphics Forum 23*, 3, 381–389.

VANAKITTISTIEN, N., SUDSANG, A., AND CHENTANEZ, N. 2016. 3d hair model from small set of images. In *Proceedings of MIG*, ACM, New York, NY, USA, 85–90.

WANG, L., YU, Y., ZHOU, K., AND GUO, B. 2009. Example-based hair geometry synthesis. *ACM Trans. Graph. 28*, 3 (July), 56:1–56:9.

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 Trans. Vis. Comp. Graph. 13*, 2, 213–234.

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

WEI, Y., OFEK, E., QUAN, L., AND SHUM, H.-Y. 2005. Modeling hair from multiple views. *ACM Trans. Graph. 24*, 3 (July), 816–820.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE TPAMI 29*, 3, 463–476.

YU, X., YU, Z., CHEN, X., AND YU, J. 2014. A hybrid image-cad based system for modeling realistic hairstyles. In *I3D '14*, ACM, 63–70.

YUKSEL, C., SCHAEFER, S., AND KEYSER, J. 2009. Hair meshes. *ACM Trans. Graph. 28*, 5, 166.

ZHAO, H. 2005. A fast sweeping method for eikonal equations. *Mathematics of computation 74*, 250, 603–627.

Appendix

Symbol	Description
I_i	an input hair image
T_i	the camera parameters for I_i
M_i^I / M_i^H	a hair mask
D^I	a distance field
$A(\cdot)$	computes the area of a mask
G	a 3D hair shape
$P_d(\cdot)$	projects a vector along a direction d
W_i	a thin-plate-spline-based warping function
δ	the magnitude of Laplacian coordinates
p_j / q	a point on G
I	the hair texture
$\bar{I}_i$	a synthesized texture using I_i as the source
$\bar{I}$	the texture after blending $\{\bar{I}_i\}$ using weights $\{\alpha_i\}$
∇I	the gradient of I
S / R	the source / target region in texture synthesis
X / Y / Z	an $m \times m$ patch
x	a texel in a patch
$C(\cdot)$	returns the center point of a patch
$U(\cdot)$	returns the uv coordinates of a point
$N(\cdot)$	computes the local neighborhood of a point
$f(\cdot)$ / $\hat{T}_k$	an affine transformation
g_j	the NNF at p_j
u / r	a 2D random offset / variable
w	a maximum 2D search distance
t	2D texture coordinates
γ	a weight for color voting

Table 1: *Summary of the notation used in hair texture synthesis.*

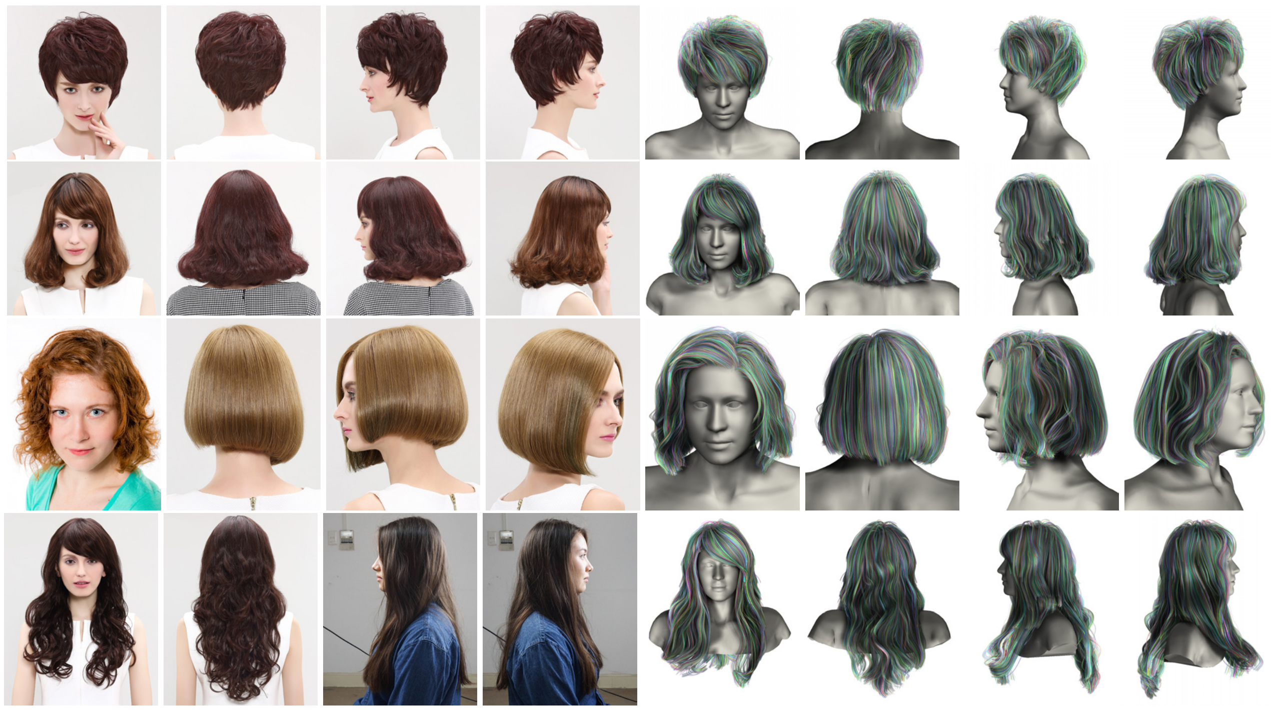


Figure 14: *Modeling with images that depict similar but not exactly the same hair (the top two rows), and even with images of considerably different hairs in different views (the bottom two rows). The input images are shown on the left, and our modeling results are on the right.*

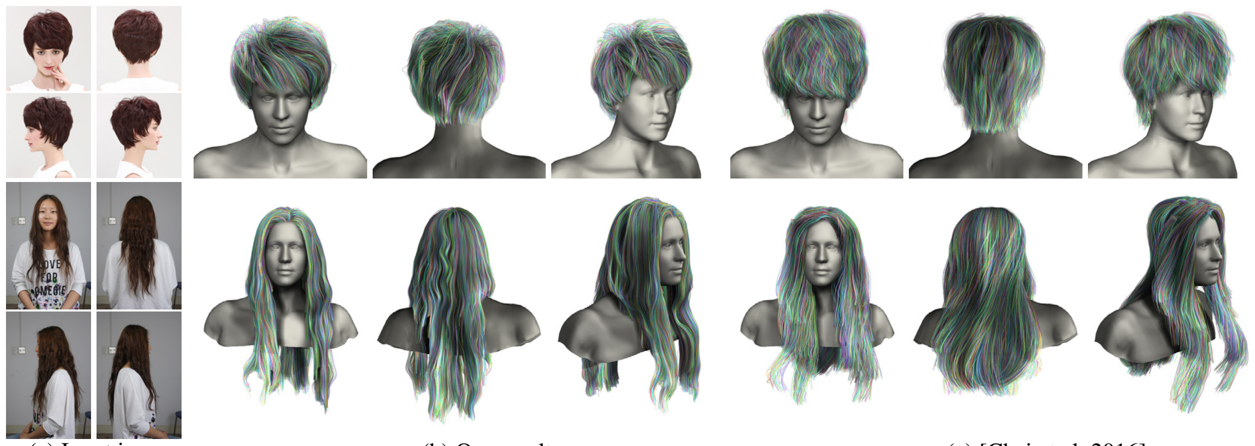

(a) Input images (b) Our results (c) [Chai et al. 2016]

Figure 15: *Comparisons with a state-of-the-art single-view modeling method [Chai et al. 2016]. For left to right, our input images, our results and the results using [Chai et al. 2016].*

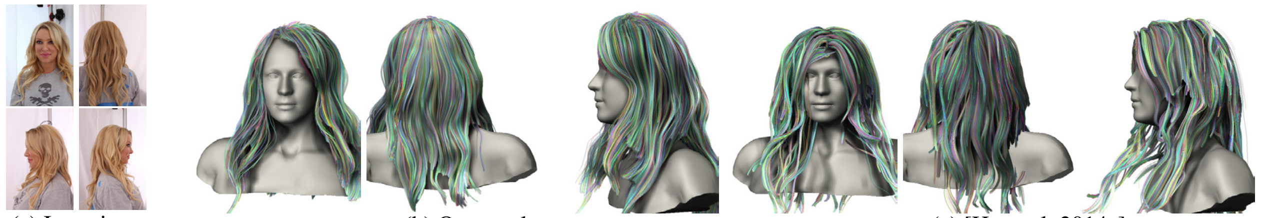

(a) Input images (b) Our results (c) [Hu et al. 2014a]

Figure 16: *Comparisons with a state-of-the-art multi-view modeling approach [Hu et al. 2014a]. For left to right, our input images, our results, and the results of [Hu et al. 2014a] using 66 input photos.*

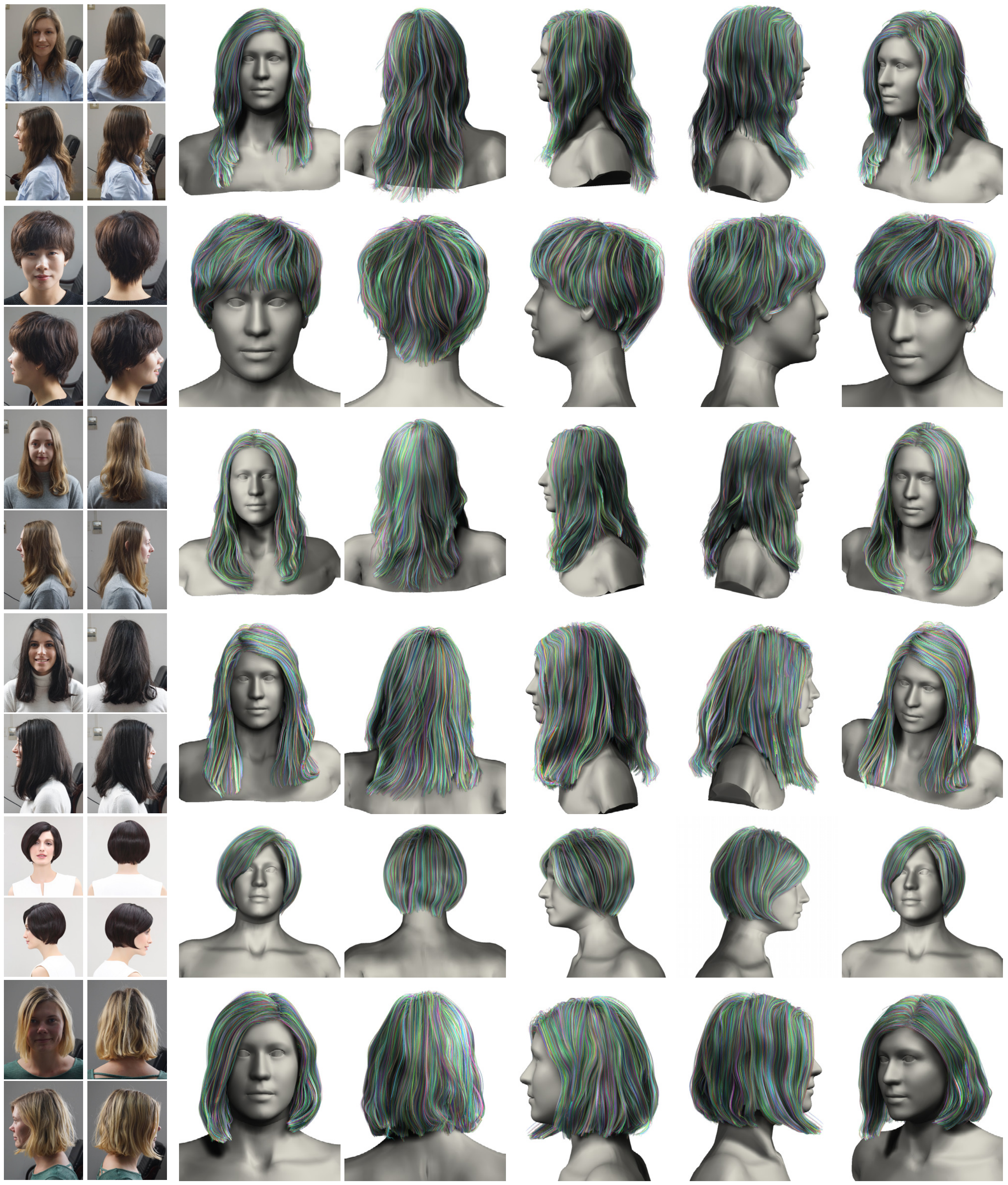


Figure 17: *Our hair modeling results with four-view images. For each row, the input images are shown on the left, and our modeling results are on the right.*