DeepSketchHair: Deep Sketch-based 3D Hair Modeling

Yuefan Shen[1] Changgeng Zhang[1] Hongbo Fu Kun Zhou[†] Youyi Zheng[†]

Abstract—We present *DeepSketchHair*, a deep learning based tool for modeling of 3D hair from 2D sketches. Given a 3D bust model as reference, our sketching system takes as input a user-drawn sketch (consisting of hair contour and a few strokes indicating the hair growing direction within a hair region), and automatically generates a 3D hair model, which matches the input sketch both globally and locally. The key enablers of our system are two carefully designed neural networks, namely, *S2ONet*, which converts an input sketch to a dense 2D hair orientation field; and *O2VNet*, which maps the 2D orientation field to a 3D vector field. Our system also supports hair editing with additional sketches in new views. This is enabled by another deep neural network, *V2VNet*, which updates the 3D vector field with respect to the new sketches. All the three networks are trained with synthetic data generated from a 3D hairstyle database. We demonstrate the effectiveness and expressiveness of our tool using a variety of hairstyles and also compare our method with prior art.

Index Terms—Sketch-based hair modeling, 3D volumetric structure, deep learning, generative adversarial networks

1 Introduction

The development of digital entertainment leads to the growing demand for 3D content. 3D virtual human plays a critical role in the 3D digital world, while 3D hair is probably the most challenging part of a human body to model. Different from other parts of the human body that can be well modeled as surfaces, the hair is usually modeled as strands, reflecting extreme variability and geometry complexity. What's more, mutual occlusion among strands increases the difficulty of hair modeling.

Image-based hair modeling has gained substantial attention in recent years. It is now possible to reconstruct realistic and high-quality hairstyles from either multi-view images [1], [2], single-view images [3], [4], [5], or RGB-D images [6]. Lately, deep learning based approaches [7], [8], [9] have also been exploited for modeling 3D hair from images. Although promising results have been achieved with these methods, one major downside in them is that hairstyles are extracted from images rather than free-form designed. For example, all existing image-based techniques focus on extracting unabridged 3D hair from portrait images, and there is no interface provided for users to freely modify reconstructed hair models. For consumer-level applications, it is highly desirable to have a system that can not only model hair from images or even from scratch but also be capable of performing efficient free-form edits, to create user-desired hairstyles.

There are a few prior works focusing on interactive modeling of 3D hairstyles from sketches. Earlier attempts [10], [11], [12] produce hair models made up of lots of wisps by heavy and complex interaction. The methods of [13], [14], [15] aim to simplify interactive modeling of strand-level hair, which can be achieved by 2D sketches. However, their generated hairstyles do not own high quality or complex geometry. [16], [17] explore vector field based methods and use 3D strokes to drive a 3D vector field which represents the growing direction of hair strands, but authoring 3D curves is not an easy task for novice users and it is not intuitive for users to imagine a resulting hairstyle from a vector field. There also exist professional interactive hair modeling systems such as Maya, SolidWorks, 3D Max, CATIA, which incorporate powerful tools for accurate and detailed geometric hair model construction and manipulation. However, to guarantee the quality of produced models, these systems sacrifice usability and often require highly trained skills. Recently, Xing et al. [18] proposed a hair modeling method, which allows easy creation of high-quality strand models by interactively authoring 3D curves in VR, followed by partial retrieval. However, such immersive interaction requires head-mounted devices and additional hardware setups.

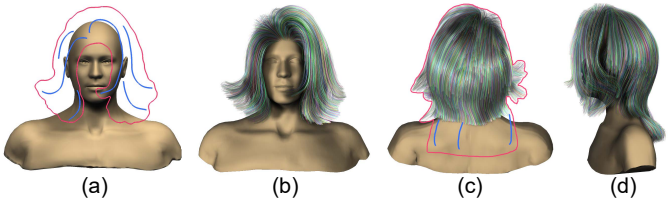


Fig. 1: Our system, *DeepSketchHair*, takes as input a 2D sketch (a) consisting of a hair contour (red) and a few strokes (blue) indicating the hair growing directions, and generates a realistic 3D hairstyle (b). Users can continue to modify the generated hair model from another view with a new sketch (c), and our system updates the result accordingly (d).

We introduce *DeepSketchHair*, a deep learning based system for easy and intuitive modeling of high-quality strand-level hairstyles from 2D sketches. As illustrated in Fig. 2, our pipeline starts with a sketched hair contour, which is easily converted to a mask map corresponding to the hair region of a desired 3D hairstyle, and additional strokes as a sketch strand map for

[1] *co-first authors*
[†] *corresponding authors*

- *Y. Shen, C. Zhang, K. Zhou, and Y. Zheng are with the State Key Lab of CAD&CG, Zhejiang University, China. E-mail: zyy@cad.zju.edu.cn*
- *H. Fu is with the School of Creative Media, City University of Hong Kong.*

rough specification of sparsely growing direction of the hair. Then *S2ONet*, a generative adversarial network (GAN) model for pixel-to-pixel translation [19], is adopted to predict a dense 2D orientation field, in which each pixel stores the direction of a 2D hair strand passing through it. Next, *O2VNet*, another GAN-based network, converts the 2D orientation field to a 3D vector field[1]. To account for the lack of depth information, we add the depth map of a 3D bust model as a 3D guidance for *O2VNet*. Then, high-quality strand-level hairs are grown from the 3D orientation field. *DeepSketchHair* allows users to modify the resulting 3D hairstyle by inputting additional 2D sketches in new views. Our generated hair models can be free-form adjusted with the help of *V2VNet*, which is also based on GAN but this time for voxel-to-voxel conversion to update the 3D vector field w.r.t. the user input under a new view. Lastly, we provide user-friendly auxiliary tools for fine-tuning the generated hairstyles.

We demonstrate the effectiveness and expressiveness of our modeling system by creating various hairstyles, spanning a diverse spectrum of hair shapes, such as smooth, jagged, wavy, or curly. Our system owns a powerful ability to generate these complex hairstyles, as seen in Fig. 1 and Fig. 11. In summary, our main contributions are listed as follows:

- A deep learning based framework for sketch-based hair modeling, which can generate high-quality strand-level 3D hairstyles from 2D sketches;
- Three novel generative adversarial neural networks, namely, *S2ONet*, *O2VNet*, and *V2VNet*, to collaboratively support multi-view hair modeling.

2 Related Work

Our work tackles the problem of sketch-based hair modeling using deep learning. Below we review existing works closely related to our method, namely, those in hair modeling and sketch-based modeling.

Hair Modeling. Modeling 3D hair strands is a difficult task due to the variety of hairstyles and the complex structures of internal hair. Early works in hair modeling start from scratch while recent works focus more on image-based hair reconstruction. We refer readers to [20] and [21] for comprehensive surveys on hair modeling.

Watanabe et al. [10] introduce a wisp model for 3D hair and this model has been widely applied in early works for interactive hair modeling. Some of these methods directly model hair strands cluster by cluster with drawn 3D curves e.g. [22], [23], [24]. Other approaches allow users to model a hairstyle roughly first and then solve wisp-based deformation by interaction [11], [12]. Normally, it is a time-consuming task to model a realistic hairstyle with these methods, and trained skills are required.

To make the interaction easier, in [14] Malik presents a sketching interface for modeling and editing hairstyles by controlling the generation and deformation of hair clusters. Wither et al. [15] explore a physically-based approach to infer physically-based parameters from 2D user sketches for hair modeling. Using sketching interfaces makes the modeling process easier and faster, but the quality of generated hairstyles is not very high. Using VR equipment, the method in [18] allows easy creation of 3D curves for partial hairstyle retrieval, which can model realistic hair strands. Yet, the interaction is restricted in immersive environments with head-mounted devices.

A vector field based solution [16] can reduce the time overhead of many manual operations and supports complex hair modeling by initializing or editing the underlying vector fields. One limitation of this method, however, is that vector fields are not very intuitive for interactive editing. Fu et al. [17] present a method that allows users to edit a vector field by setting various constraints using 3D curves. However, authoring 3D curves is not an easy task for novice users and there is still a gap between input curves and generated results.

To reduce the complicated manipulation and strengthen the bond between the input and generated hair, multiple image-based hair modeling methods have been proposed. Ming et al. [25] were probably the first to use real images from different views to generate 3D hair models. Their method builds a 3D volume for hair generation using extracted information from images, like hair outline and hair flow direction. However, this method is only suitable for modeling simple hairstyles. By accurately detecting hair orientation from images, image-based hair modeling is able to generate complex and realistic 3D hair models. Paris et al. [26] use various viewpoints as well as several oriented filters for hair orientation detection in different local parts. Wei et al. [27] extend image-based hair modeling to a more flexible level, and their method does not rely on special capturing setup under controlled illumination conditions [26]. More recently, Zhang et al. [1] propose a method which takes only four-view images to generate a 3D hair model using a predefined hair database. Meanwhile, a few attempts have been made to achieve single-view image-based hair modeling. For example, [3] and [28] rely on the human annotation to generate a hairstyle from a single portrait image. Hu et al. [5] propose a method based on database retrieval for realistic 3D hair modeling. Single-view based methods can only generate 3D hair models matching the input view and lack control over hair geometry from new views.

Deep learning techniques have been exploited for single image-based hair modeling in recent years. Chai et al. [29] introduce a deep convolutional network for segmenting hair regions and estimating hair growth orientation, which guides their data-driven hair modeling method. A similar idea is adopted by Hu et al. [2] for hair digitization, which uses neural networks for hair classification and region segmentation. Zhou et al. [7] present an encoder-decoder network architecture to generate strand features from 2D orientation input for hair growing. Saito et al. [8] extend this idea by adopting a variational auto-encoder structure [30] to encode the hair information and image information into one latent space. Lately, Zhang and Zheng [9] propose Hair-GAN, which uses a GAN structure with 3D convolutional layers to generate 3D orientation fields for hair growing. Different from the above approaches, we consider modeling 3D hair from casual 2D sketches. Unlike images that contain rich information, sketches are inherently sparse and ambiguous, which could pose significant challenges for network learning [31].

Sketch-based Modeling. Sketching is one of the most intuitive and easiest ways for users to interact with computers. However, it remains difficult for computers to interpret freehand sketches due to their inherent sparsity and ambiguity. This leads to a thread of sketch-based modeling methods.

Early free-form modeling methods create 3D contents from user-drawn 2D sketches using inflation (e.g. [32], [33]). Although inflation is unambiguous, it can only produce rough 3D surface

1. We refer to a 3D orientation field as a 3D vector field here to emphasize its difference from a 2D orientation field.

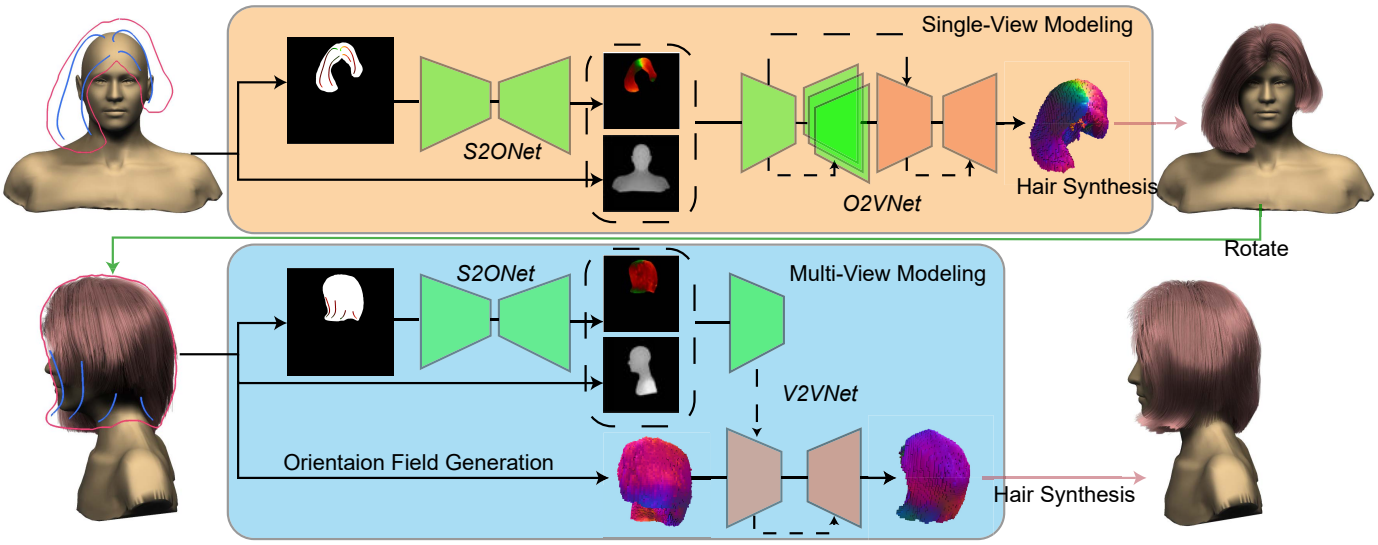


Fig. 2: The pipeline of our *DeepSketchHair* framework. The single-view modeling module (top tow) takes as input a user-drawn sketch on top of a bust model (top left), and generates a hair strand model (top right) from a synthesized intermediate 3D orientation field. The multi-view modeling module (bottom row) takes as input the bust model, the currently synthesized hairstyle, and an additional sketch in a rotated view, and generates the updated strand model (bottom right). Our three generative adversarial networks (*S2ONet*, *O2VNet*, and *V2VNet*) are shown in the middle.

models in simple shapes without self-intersections. Further efforts have been dedicated to edit generated rough models by drawing constrain curves like [34]. With strong priors, geometry fitting based approaches retrieve the most similar 3D geometry structures of 2D sketches part by part [35], [36]. Yet, the complexity of their modeling results is limited by simple primitives.

In recent years, deep learning based techniques have also been utilized to learn a desired mapping from 2D sketches to 3D features. Huang et al. [37] introduce deep convolutional neural network (CNN) for mapping 2D sketches to procedural model parameters. Delanoy et al. [38] propose an end-to-end CNN, which is trained to generate 3D models from 2D multi-view sketches. In [31], Li et al. learn 2D middle maps to guide robust 3D modeling from 2D sketches. Lun et al. [39] generate depth maps from multi-view sketches and then fuse them for 3D reconstruction. Su et al. [40] learn normal maps directly from 2D sketches while Han et al. [41] utilize deep neural networks to predict latent code of faces from 2D sketches for the generation of detailed face models.

Unlike the aforementioned sketch-based modeling methods, which aim to infer 3D surfaces from 2D sketches, our task requires the generation of 3D hair strands exhibiting varying attributes, and thus cannot be solved directly by applying existing sketch-based modeling methods. To the best of our knowledge, our work is the first deep learning based technique for inferring strand-level 3D hairstyles from 2D sketches.

3 Overview

Fig. 2 shows our deep learning based hair modeling pipeline. Given a bust model, the user sketches over it to scribble a hair mask (the contour, shown in red) and hair strand orientation strokes (shown in blue). We require the user to draw directed strokes to mimic the strand growing direction such that the resulting sketch map is oriented and unambiguous (the tangent direction serves as the orientation at a specific stroke point). The backbone of our system consists of three deep neural networks independently aggregating information to support sketch-based hair modeling. The first network is named *S2ONet* (§4.1), which converts the hair mask and the sketch map to a dense 2D orientation field. The second network is named *O2VNet* (§4.2), which maps the dense 2D orientation field to a 3D volumetric vector field, with the help of the depth map of the bust model. The resulting 3D vector field is then used to grow hair strands for hair synthesis (§5.3).

Since using only one view of sketches might not be sufficient to generate user-desired 3D hair strands, our approach allows users to adjust the currently synthesized hair strands by drawing additional sketches or masks in new views (§5). The third network, named *V2VNet* (§5.1), then updates the volumetric 3D vector field with respect to the new inputs. More specifically, a newly drawn sketch is firstly converted to a dense 2D orientation field (by *S2ONet*), which is then fed into *V2VNet* together with the rotated 3D orientation field produced in the previous view to generate the updated 3D orientation field in the current view. The hair strands are updated according to the new orientation field. Users can repeatedly edit the hair strands by sketching in any views.

As complementary tools, auxiliary hair editing operations are added to support hair refinement (§6.1). The auxiliary operations include cut, deform, color change, enlength, and so on.

4 Single-View Hair Modeling

Given a sketch image I_S and a mask image I_M (Fig. 2), our goal is to generate a corresponding 3D vector field or orientation volume $\mathcal{Y}$ to grow realistic hair strands. One primary challenge here is to generate a desired dense 3D orientation field from the 2D sparse and inherently ambiguous input. As shown in [31], a direct learning of 3D information from sparse 2D sketches could easily lead to undesirable results. This motivated us to add an intermediate representation, namely, a dense 2D orientation field,

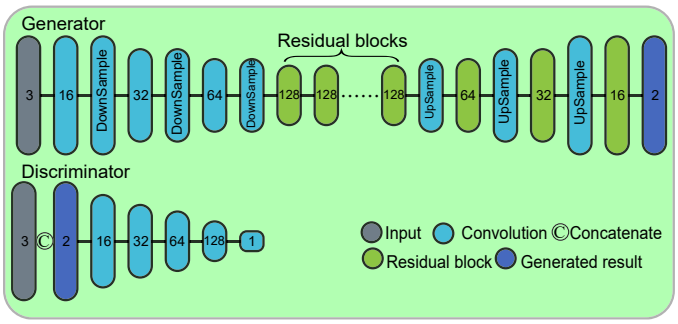


Fig. 3: Architecture of our *S2ONet*. For an input pair of sketch image and mask image $\{I_S, I_M\}$, we enclasp them into a 2D tensor of size $128 \times 128 \times 3$. For the generator, the contracting part has 3 downsampling convolutional modules with $(32, 64, 128)$ output channels and the expanding part has 3 upsampling deconvolutional modules with $(64, 32, 16)$ output channels connected with 8 residual blocks. The input of our discriminator concatenates the generated *fake* 2D orientation map Λ or *real* map $\widetilde{\Lambda}$ with the input condition $\{I_S, I_M\}$, which passes through 4 convolutional layers and 1 full-connected linear layer.

between the sketch domain and the volumetric domain to better bridge the gap.

In this section, we focus on our single-view hair modeling framework, which consists of two main parts: *S2ONet* and *O2VNet*, as illustrated in Fig. 2 (Top). Taking I_S and I_M as input, we first generate a dense 2D orientation field Λ with *S2ONet*. Next, Λ and the bust depth image $\mathcal{D}$ are taken as the input to generate $\mathcal{Y}$ with *O2VNet*. Both networks use a conditional GAN structure [42].

4.1 *S2ONet*

We apply WGAN-GP [43] for the dense 2D orientation field generation. For this task, the input of the network is a 2D tensor with a size of 128×128, including 3 feature channels: 2 channels of oriented sketch map and 1 channel of binary mask map. Specifically, the sketch orientation map uses R and G channels in RGB to represent a 2D direction vector (x, y) (per stroke point and linear mapped from values of [-1,1] to values of [0, 255]). The output is also a 2D tensor including 2 feature channels, which represents 2D direction vectors (x, y) of the dense 2D orientation field Λ.

Network Architecture. Fig. 3 describes the generator and the discriminator of our GAN network. We use an encoder-decoder structure in the network similar to approaches for generating high-resolution images with low-resolution inputs [44], [45]. Like [44], our generator consists of residual blocks, upsampling blocks, and downsampling blocks. We also connect the encoder and decoder with residual blocks.

Since our target dense 2D orientation fields are more unified and simpler than colorful pictures, to avoid overfitting, we simplify the network in [44] by reducing half number of filters in all convolutional layers. In addition, our discriminator is composed of five convolutional layers and one linear layer, all without dropout, and takes the corresponding sketch and mask maps as conditional inputs. Specifically, we concatenate the generated fake results or real target with the input sketch and mask images as condition to generate a 2D input tensor with a size of $128 \times 128 \times 5$.

Loss Function. With a sketch image I_S and a mask image I_M as input, we want that our *S2ONet* generator's result $\Lambda = G_s(I_S, I_M)$ gets close to the target $\widetilde{\Lambda}$. Previous works on conditional image generation show the benefit of the feature loss, which is defined as the $L2$ difference in a feature space extracted from a middle layer of a pre-trained network [44]. However, most of pre-trained networks work on classification or segmentation of photographs while our target images are orientation maps, which are very different from photographs.

Instead of using the traditional adversarial loss and feature loss [19] for the generator, which cannot fully describe the differences between the real and fake distributions, we adopt a similar idea in [9], [46] to use a combination of content loss and style loss for our task. Our *S2ONet* generator loss is defined as:

$$\mathcal{L}_{G_s} = \alpha \sum_{l \in m} \mathcal{L}_{content}(l) + \beta \sum_{l \in n} \mathcal{L}_{style}(l). \tag{1}$$

Here, the first term is the content loss and the second one is the style loss. l denotes the layer index. We set $\alpha = 0.01$ and $\beta = 5$. We select the middle layers $m = \{0, 2\}$ of the discriminator to compute the content loss and middle layers $n = \{0, 1, 2, 3, 4\}$ to compute the style loss.

$$\mathcal{L}_{content}(l) = \frac{1}{2} \sum_{i,j} [F_{l,i,j}(\Lambda, I_S, I_M) - F_{l,i,j}(\widetilde{\Lambda}, I_S, I_M)]^2. \tag{2}$$

In Eq. (2), $F_{l,i,j}$ represents the discriminator features of the i^{th} filter at position j in the discriminator layer l. Note that if $l = 0$, we set $L_{content}(0) = \frac{1}{2}[\Lambda - \widetilde{\Lambda}]^2$, which means the per-pixel loss between generated results and the ground truth. Based on [46], we define our style loss as follows:

$$\mathcal{L}_{style}(l) = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} [A_{l,i,j}(\Lambda, I_S, I_M) - A_{l,i,j}(\widetilde{\Lambda}, I_S, I_M)]^2. \tag{3}$$

In Eq. (3), A_l is the Gram matrices, where $A_{l,i,j}$ is the inner product between the vectorized feature maps i and j in the l^{th} layer: $A_{l,i,j} = \sum_k F_{l,i,k} F_{l,j,k}$. N_l is the number of feature channels and M_l is the total size of feature tensors.

The loss of the discriminator is defined as:

$$\mathcal{L}_{D_s} = D_s(\widetilde{\Lambda}, I_S, I_M) - D_s(\Lambda, I_S, I_M) + \lambda \mathcal{L}_{gp}, \tag{4}$$

where

$$\mathcal{L}_{gp} = (\|\nabla_{\hat{\Lambda}} D_s(\hat{\Lambda}, I_S, I_M)\|_2 - 1)^2. \tag{5}$$

$D_s(\mathcal{X})$ represents the *S2ONet* discriminator. $\mathcal{L}_{gp}$ is the gradient penalty for the random sample $\hat{\Lambda}$ proposed in [43], where $\hat{\Lambda} = \epsilon\widetilde{\Lambda} + (1 - \epsilon)\Lambda$, and ϵ is a random number $\epsilon \sim [0, 1]$. The value of λ is set to 10.

4.2 *O2VNet*

After *S2ONet*, we get a dense 2D orientation field and our *O2VNet* is designed to predict a voxel-represented 3D orientation field from this 2D input using a GAN structure. Previous works showed that the GAN with 3D convolutional layers does well in generating voxels [47] and has been adopted by Zhang and Zheng [9] for image-based hair modeling. In our case, the input of *O2VNet* generator is a 2D tensor with a size of $128 \times 128 \times 3$, including a 2D orientation field Λ and a 2.5D bust depth image $\mathcal{D}$. The output is a 3D tensor $\mathcal{Y}$ of size 128 x 128 x 96, representing a 3D orientation field.

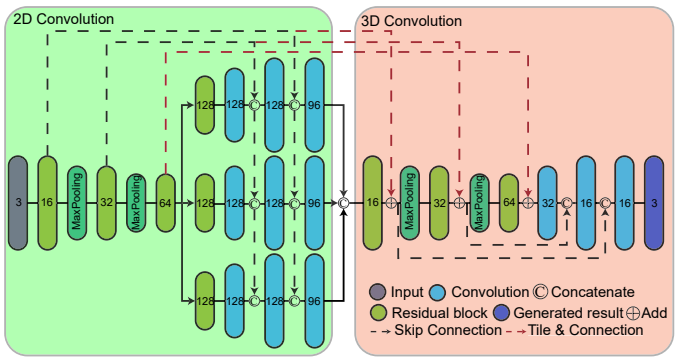


Fig. 4: Architecture of our *O2VNet* generator, which takes the input $\{\Lambda, \mathcal{D}\}$ with the shape of $128 \times 128 \times 3$. The encoder of 2D convolutional part takes three residual blocks with $(16, 32, 64)$ output channels combing with two max pooling layers for downsampling, followed by three decoders, each of which contains one residual block and three deconvolutional layers all with 128 output channels for upsampling. The outputs of three decoders are concatenated and reshaped to a 3D tensor with the size of $128 \times 128 \times 96 \times 3$. The 3D convolutional part resembles a 3D U-Net structure, which includes an encoder with two downsampling modules with $(32, 64)$ output channels and a decoder with two upsampling modules with $(32, 16)$ output channels. The skip connection and our tile & connection are described in different colors (black and red), respectively.

Network Architecture. As shown in Fig. 4, the generator of *O2VNet* is constituted of two main modules: a 2D convolutional module and a 3D convolutional module, each with an encoder-decoder structure based on the commonly used U-Net [48].

The 2D convolutional module (Fig. 4 (Left)) is composed of an encoder and three decoders of the same shape for predicting values at different axes X, Y, Z. The output of three decoders are three 2D tensors of size $128 \times 128 \times 96$, which can be concatenated and reshaped into a 3D tensor of size $128 \times 128 \times 96 \times 3$. To make filters in our encoder have larger receptive fields of input images and thus learn better occupancy features (see comparisons in §7.3), we set layers of residual blocks all followed by max-pooling layers.

Our 3D convolutional module consists of two downsampling modules with residual blocks and max pooling layers, and two upsampling modules. We add the 2D feature maps to the 3D feature maps to ensure that the 2D detail information shall go through the whole network. To to this, we tile 2D feature maps to the size of 3D feature maps before performing the feature addition. We call this step *tile connection*. We share the same discriminator architecture with [9] which consists of 5 layers of 3D convolutional layers.

Our architecture differs from *HairGAN* [9] in the following aspects. First, we use the U-Net structure for both the 2D convolution and 3D convolution to ensure that the high-frequency features can flow through the network whereas HairGAN only exploits downsampling modules without skip connection. Second, we use *tile connection* to let the details further flow from the 2D module to the 3D module. Two two steps ensures that the details originally in the 2D orientation maps are preserved in the results (see §7.3). Third, our 3D convolutional module has more 3D convolutional layers than the simple 3D convolutional layers used in [9] and thus merits better model capacity.

Loss Function. Like *S2ONet*, *O2VNet* applies WGAN-GP to speed up the convergence and defines the generator loss as a combination of the content loss and the style loss. In addition, we hope that the projection of the generated 3D orientation field from the input view is as similar as possible to the 2D orientation field to synthesize user-desired hair strands. Thus, we add the projection loss to *O2VNet*:

$$\mathcal{L}_{proj} = \sum_{i \in I_\Theta} (Proj(\mathcal{Y})^i - \Lambda_i)^2. \tag{6}$$

Here, I_Θ represents the 2D region where the 2D orientation field has valid values in the 2D image and $Proj(\mathcal{Y})$ is projected 2D region of the 3D orientation field $\mathcal{Y}$ (from visible voxels). The projection part in our approach differs from existing works, which commonly use a differentiable projection layer for end-to-end training [49]. In our case, the camera view of projection is fixed in the training step, and the cross section of the generated voxel shares the same solution as the input 2D orientation field (both are of the size 128×128). Hence our projection matrix P can be fixed. Let $\mathcal{Y}'$ denote the projected 3D orientation field $P(\mathcal{Y})$. We store the indices of the visible voxels of $\mathcal{Y}'$ in advance using its ground truth $\widetilde{\mathcal{Y}}$, and use the indices to directly compute the values of $Proj(\mathcal{Y})$. In particular, we simply use the normalized x and y parts of $\mathcal{Y}'$ while ignoring the z part.

In our experiments, we find that adding the projection loss could only constrain the voxels in the visible area. To diffuse the constraints to the whole 3D orientation volume, we add the following Laplacian loss:

$$\mathcal{L}_{lap} = \sum_i (\Delta(\mathcal{Y}_i) - \Delta(\widetilde{\mathcal{Y}}_i))^2, \tag{7}$$

with

$$\Delta(\mathcal{Y}_i) = \sum_{j \in N_i} \frac{1}{|N_i|}(v_j - v_i). \tag{8}$$

Our Laplacian loss represents the difference of the divergence of the gradient between the generated 3D orientation field and the ground truth $\widetilde{\mathcal{Y}}$. $\Delta(\mathcal{X})$ is the Laplace operator and i in Eq. 7 represents each voxel in the 3D orientation field. In Eq. 8, N_i is the index set of the neighbors of the voxel $\mathcal{Y}_i$, and v_i, and v_j are the values of the voxels $\mathcal{Y}_i$ and $\mathcal{Y}_j$, respectively.

Our final generator loss is defined as:

$$\mathcal{L}_{G_d} = \iota\mathcal{L}_{content} + \kappa\mathcal{L}_{style} + \gamma\mathcal{L}_{proj} + \varepsilon\mathcal{L}_{lap}. \tag{9}$$

In our experiments, $\iota = 0.01$, $\kappa = 5$, $\gamma = 0.1$, and $\varepsilon = 2e - 5$. Note we do not normalize each loss.

4.3 Data Generation

To train our networks, we prepared a hair dataset, which is partly from public online repositories [8] and partly provided by [29]. We collected totally 653 3D strand-level models, and aligned each hair model to a unified bust model within a volumetric bounding box. We fixed the camera at the front of the box and placed a view-orthogonal sketch plane in the middle of the box.

For our single-view modeling, we project each pair of hair and bust models to the front view to generate the rendering of the corresponding ground-truth 2D orientation map, mask map, and bust depth map. We randomly rotate the hair-strand model and bust model together around the box center to augment the training dataset. The rotation ranges from $+30^o$ to -30^o for the Y axis

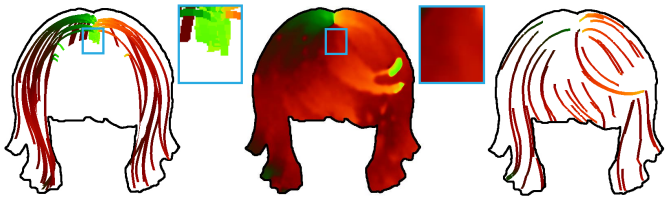


Fig. 5: From left to right: the sketch map generated by rendering randomly selected hair strands, the corresponding ground-truth 2D orientation map, and the sketch map generated with our image tracing method. The rendered result (Left) may include orientation information from invisible hair strands, and does not match with the GT 2D orientation map well. In additional, a random selection may lead to non-uniform distribution.

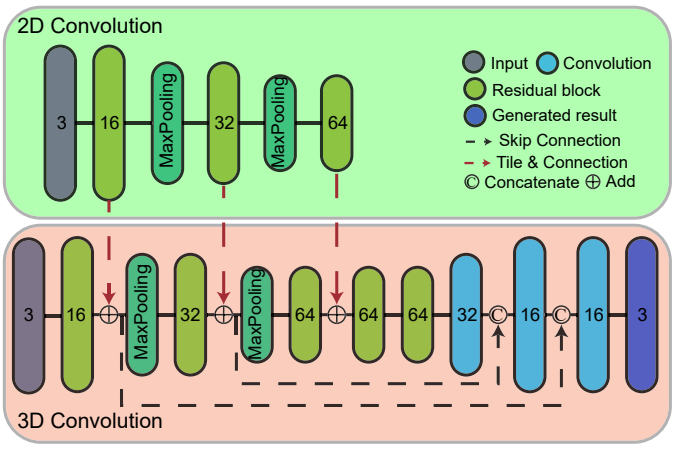


Fig. 6: The generator architecture of our *V2VNet*. The structures of 2D and 3D convolution are similar to that of *O2VNet*.

and $+15^o$ to -15^o for both the X and Z axes. X, Y, and Z are the world coordinate axes. We describe some details in the followings.

2D Hair Maps. To generate dense 2D orientation maps, we render hair strands into images with every vertex's color of strand representing its tangent direction. We use OpenGL depth test to avoid rendering of invisible strands. Similarly, we generate hair mask maps by replacing every strand vertex's color in rendered orientation maps by white. To generate the depth map of the bust model, we cast a ray parallel to the Z axis, assuming the ray intersects the bust in p and the z-component value of p is p^z. We define b_{min} as the box minimum point whose depth value b^z_{min} is the minimum among all box points. The box maximum point b_{max} and its depth value b^z_{max} are similarly defined. Then the normalized depth of the current ray intersection is calculated as $\frac{p^z - b^z_{min}}{b^z_{max} - b^z_{min}}$. Finally, we get the dense orientation map Λ, the mask map I_M, and the bust depth map $\mathcal{D}$. All of the hair maps have the resolution of 128×128 in our implementation. Unlike [9], which uses a high resolution image (1024×1024) and then performs downsampling to 128×128, we use a direct 128×128 input for two reasons. On one hand, such a resolution is in conjunction with the fused 3D volume and in our experiments we find it to be sufficiently informative to synthesize vivid 3D orientation fields for hair synthesis (see comparisons to [9] in §7); On the other hand, synthesizing high resolution images with high quality is known to be difficult for a GAN model such as our *S2ONet* [50].

2D Sketch Map. A direct way to generate a sparse sketch map for training *S2ONet* is to randomly sample hair strands from an entire hair model. However, one might note that the dense 2D orientation map we wish to synthesize only depicts the hair orientation of visible strands. Sampling from invisible hair strands could lead to ambiguities in the input, as shown in Fig. 5 (Left). To avoid such ambiguities, we generate the sketch map by tracing the dense 2D orientation map. Specifically, we first randomly select a seed pixel on the dense orientation map, and trace from this seed pixel by iteratively finding the next pixel among eight neighboring pixels with the smallest color discrepancy. Supposing the current pixel value is $p(x, y)$ (representing the projected strand orientation at this position) and the value for one of its neighbors is $p^n(x, y)$, we mark p_n as a candidate pixel if the dot product between $p(x, y)$ and $p^n(x, y)$ is below a given threshold ($\varrho = 0.5$ in our case). After checking all eight neighboring pixels of the selected pixel, we choose the most matching pixel among the candidate pixels as the next iterative pixel by finding the minimum dot product. If there is no candidate pixel, we terminate the tracing process and randomly re-select a new seed pixel for the next tracing process. Finally, we get the sketch map I_S (Fig. 5 (Right)) through multiple iterations. To ensure uniform sampling, we employ an adaptive clustering strategy as in [51] and then select one strand in each cluster.

3D Hair Orientation Field. As in [9], we convert hair strands to a 3D orientation field $\mathcal{Y}$. We define a grid volume with the size of $128 \times 128 \times 96$. Every grid cell stores a value which is the average tangent direction of hair strands that pass through this grid cell.

5 Multi-view Hair Modeling

In our system we allow users to draw a new sketch or mask to adjust the hair model in either the current view or a novel view. This is fully supported by our multi-view hair modeling module. This module takes as input the 3D orientation field $\mathcal{Y}$ (after rotation), the bust depth map (after rotation), and the newly specified sketch and hair mask. We first generate a dense 2D orientation field Λ^* from the new sketch and mask images using *S2ONet*. The system then outputs a new 3D orientation field $\mathcal{Y}^*$. In this process, we hope that our network only changes the 3D orientation field in the new view with respect to the new sketch, while it can keep the original values in the other areas. Existing works on multi-view modeling like [38] simply concatenate 3D volume with 2D feature maps and feed them to a network only using 2D convolutional layers. However, through preliminary experiments, we find this structure performs badly for our task (§7). Other efforts [52], [53], [54] working on multi-view modeling encode and aggregate features from different views for 3D reconstruction, which is not suitable for our task of continuous updating of a 3D orientation field. We thus design a volume-to-volume network structure guided by 2D feature maps, named *V2VNet*.

5.1 *V2VNet* Architecture and Loss Function

The *V2VNet* architecture is similar to the 3D convolutional module in *O2VNet*, and is based on the V2V-PoseNet [55]. It takes a rotated 3D orientation field $R(\mathcal{Y})$ where $\mathcal{Y}$ is the orientation field in the previous view, a 2D orientation map Λ^* and a bust depth image $\mathcal{D}^*$ as input and generates an updated 3D orientation field $\mathcal{Y}^*$. As shown in Fig. 6, *V2VNet* has two modules. The

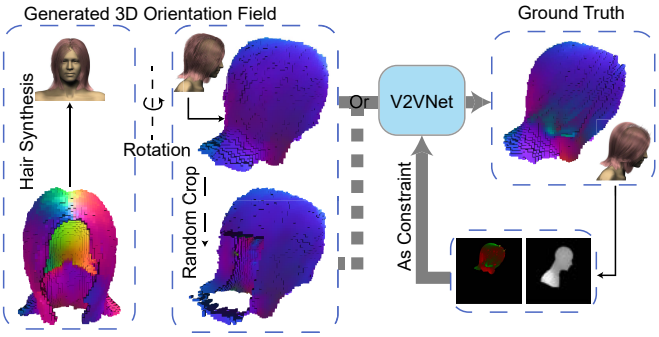


Fig. 7: Training data generation for *V2VNet*. Left: we use the generated 3D orientation field $\mathcal{Y}$ from *O2VNet* to synthesize hair strands. Middle: The rotated 3D hair model is then used for generating the rotated 3D orientation field $\mathcal{Y}^*$ as the training input for *V2VNet*. The middle bottom shows randomly cropped $\mathcal{Y}^*$. Our *V2VNet* takes $\mathcal{Y}^*$ or its cropped version as the input, and uses the 2D orientation map and bust depth map generated from the paired ground truth (Right) as constraints.

2D convolutional part takes a new 2D orientation map Λ^* and a new bust depth image $\mathcal{D}^*$ as input and applies 2D convolutional layers with 3 residual blocks and 2 downsampling layers to extract 2D feature maps for guiding the volume-to-volume generation. The 3D convolutional part uses a U-Net structure, with both the encoder and the decoder consisting of residual blocks. Unlike the 3D convolutional sub-network in the *O2VNet*, we add more residual blocks in our *V2VNet* because this task is to modify the input 3D orientation field.

Loss Function. *V2VNet* is also based on WGAN-GP, and has the same discriminator as before. Meanwhile, for the generator of *V2VNet*, we add a loss to help the network keep the original features of the pre-generated 3D orientation fields:

$$\mathcal{L}_{ori} = \sum_{i \in \Gamma} (\mathcal{Y}_i^* - R(\mathcal{Y})_i)^2. \tag{10}$$

Here, Γ represents the invisible hair voxels in the 3D orientation field from the current view. Similar to I_{Θ_1} and I_{Θ_2}, we can initialize an index set of such invisible voxels in advance. The full loss of the *V2VNet* generator is defined as:

$$\begin{aligned}\mathcal{L}_{G_v} =& \iota \mathcal{L}_{content} + \kappa \mathcal{L}_{style} + \gamma \mathcal{L}_{proj} \\ &+ \varepsilon \mathcal{L}_{lap} + \zeta \mathcal{L}_{ori}.\end{aligned} \tag{11}$$

Here the values of weight parameters $\iota, \kappa, \gamma, \varepsilon$ are same as those used in Eq. 9 and we set $\zeta = 0.1$.

5.2 Data Preparation

To train *V2VNet*, we require paired data $((\mathcal{Y}^*, \mathcal{D}^*, \Lambda^*), \mathcal{Y}^\dagger)$. Here, $\mathcal{D}^*$, Λ^*, and $\mathcal{Y}^*$ are the bust depth map, 2D orientation map, and 3D orientation field in the current view, and $\mathcal{Y}^\dagger$ is an update of $\mathcal{Y}^*$ matching the 2D orientation map Λ^*. We can obtain $\mathcal{D}^*$, Λ^*, and $\mathcal{Y}^\dagger$ from the ground truth in the current view. However, $\mathcal{Y}^*$ should denote a 3D orientation field generated in the previous view.

Profiting from our single-view hair modeling, the synthesized 3D orientation fields from our *O2VNet* is a natural fit for $\mathcal{Y}^*$. However, since simply rotating discrete grids over a continuous space would result in aliasing, we use synthesized hair strands as an intermediate representation.

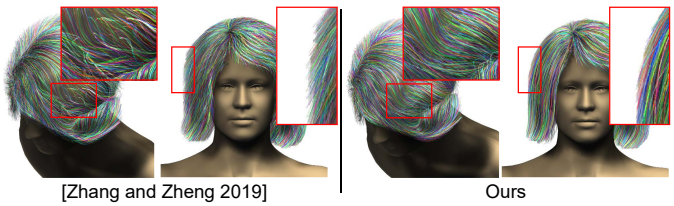


Fig. 8: Comparison of our hair growing algorithm to that by Zhang and Zheng [9] given the same 3D orientation field.

As illustrated in Fig. 7, we first use our 2D orientation map in the front view and pre-train *O2VNet* to generate a 3D orientation field $\mathcal{Y}$. We then synthesize the hair strands using the 3D orientation field $\mathcal{Y}$ (see §5.3). The hair strands are rotated and used to calculate the new 3D orientation field $\mathcal{Y}^*$. Finally, we randomly crop some parts of $\mathcal{Y}^*$ from the current view to make it have a different silhouette from the input mask. In total, there are 5,000 and 50 samples for training and validation, respectively.

5.3 Hair Synthesis

In Section 4.3, we introduce how to convert strand-level models to voxel-level models. We now introduce the method that converts voxel-level models to strand-level models. As mentioned above, we get a 3D orientation field from the output of either *O2VNet* or *V2VNet* with every grid cell storing a value, which is the average tangent direction of hair strands that pass through this cell.

Our hair growing algorithm is similar to that of [4] and [9]. We do some modifications to reduce the influence of noisy grids. First, we start from the scalp and grow hair strands from the root positions as in [4]. A strand will continue to grow along the grid cell direction to reach its next grid cell unless the next grid cell is out-of-volume or the angle difference between its direction d and the previous direction $\bar{d}$ is larger than a threshold θ. Unlike [4] and [9], which stop the growing with a small value of θ, we set θ to 150^o. More importantly, in case when θ larger than 60^o, we set the new growing direction to be the mean direction between d and $\bar{d}$ to ensure smoothness.

The above process gives us a set of "good" hair strands, denoted as S_g. We then randomly select seed grid cells inside the hair volume which have not been passed by S_g and whose value is valid (a grid value (x, y, z) is considered to be valid if $(x \times x + y \times y + z \times z) \geq 0.5$). Then we perform the growing algorithm as in [9]. This process might result in candidate hair strands S_c that may not connect to hair roots. To address this issue, for each strand $s^i \in S_c$ we try to find the most similar strand s^j from S_g, with the similarity between s^i and s^j measured in terms of their curvature. If more than $1/3$ of the continuous strand vertices of s^i whose curvature matches with that of s^j, we set s^j as the guiding hair strand for s^i and continue growing s^i using the direction of s^j and finally connect it to the neighboring roots of s^j if available. Fig. 8 shows a comparison of our hair growing algorithm to that of [9]. Note that our hair strands are more continuous and venerable in noise regions.

6 User Interface

Our system allows a user to design 3D hairstyles with or without a reference. A hairstyle can be created from scratch. In this case, the user can directly draw strokes and hair mask atop a 3D bust model

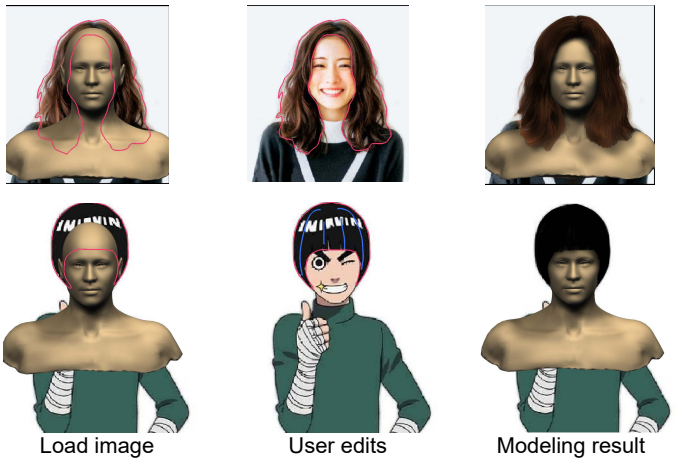


Fig. 9: The user interface. The user can load an existing portrait image as a reference or s/he can start from scratch as shown in Fig. 1. The hair masks and orientation maps can be automatically computed from the images by the method of [29].

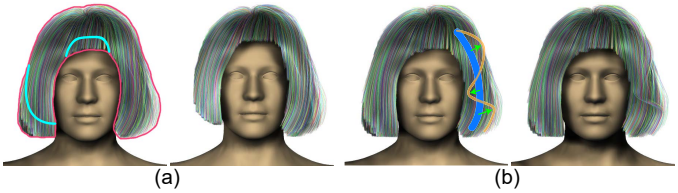


Fig. 10: Auxiliary tools ((a) cut and (b) local deform) supported in our system.

(initially in the front view). Then the user-specified sketch map and the mask map are sent to our networks to produce a 3D orientation field for hair synthesis. Alternatively, we allow the user to import a portrait image to depict a desired hairstyle (Fig. 9). In this case, our system automatically aligns the image to the bust model by using the face alignment algorithm proposed by Cao et al. [56] and segment the hair mask map by the method in [29]. The orientation map is also automatically extracted from the input image using the method described in [9] and the bust depth map is computed from the fitted bust model. In case when the user draws orientation strokes, we apply our *S2ONet* to generate the 2D orientation map instead. The generated maps are then fed into our networks to produce a 3D orientation field. Such referencing images can be used in other views as well. For multi-view modeling, the user performs the same set of operations to specify the new hair mask and strand orientations.

6.1 Hair Strand Editing

We propose some strand-level hair model editing tools for users to do simple hairstyle modifications. The supported tools include hair cutting, reshaping, lengthening(by scaling), color changing, and texturing. Below we give the details on strand cutting and reshaping.

Strand Cutting. As illustrated in Fig. 10 (a), this tool is used to cut the hair and trim the boundary of a hair strand model. The user can directly cut 3D hair strands by 2D strokes. We identify hair strands which intersect with the user-specified strokes, and discard the strands' portion which is not connected to the root. Alternatively, the user can adjust the hair strand model by editing

Index	a	b	c	d	e	f	g
views	1	1	2	2	2	2	1
strokes	0	0	0, 3	0, 3	0, 3	0, 3	10
Index	h	i	j	k	l	m	n
views	1	2	2	3	2	2	2
strokes	5	6, 3	5, 2	7, 3, 2	3, 0	4, 2	5, 2

TABLE 1: Statistics of interaction for results shown in Fig. 11. The stroke numbers are in accordance with the number of views.

the 2D hair mask. Our system monitors the 2D contour of the hair strand model and computes the new hair mask, which is then mapped to the 3D hair volume. Then we discard hair strand vertices which are outside the modified hair volume. Note when the user enlarges the hair volume by re-sketching the hair mask (i.e., not a cutting operation), we need *O2VNet* and *V2VNet* to update the 3D orientation field.

Strand Reshaping. As illustrated in Fig. 10 (b), this tool is used to deform a wisp of hair strands. Our interface allows the user to choose a wisp of hair strands in two ways. First, since a wisp of hair strand must grow from adjacent hair roots, the user can pitch on the scalp area and all hair strands whose roots are in the selected area will be selected. In the other way, the user can select a wisp of hair strands by drawing a strand sketch on the sketch plane and our system then selects hair strands whose projected shape matches the strand sketch by measuring the vertex curvatures. The selected wisp of hair strands can be deformed simply by moving the selected strand vertices and the rest of the strand vertices are updated using a method based on Laplacian editing [57].

7 Experimental Results

Our system is implemented using QT 5.12 and OpenGL. Our three networks (*S2ONet*, *O2VNet*, and *V2VNet*) are implemented using the TensorFlow framework and trained with 4 GeForce GTX 1080Ti GPUs. It took 100 epochs for *S2ONet* and 200 epochs for the other networks all with a batch size of 8 for training. Note that, for the first 100 epochs of *O2VNet* and *V2VNet* training, we do not use the projection loss or Laplacian loss. This training strategy makes the training converge faster. The whole training process took about one week. In the runtime, all of our experimental tests are conducted on a PC with an Intel(R) Core(TM) i7-8770 3.20GHz CPU and 16GB memory. Meanwhile, our system needs only one GPU for the feedforward propagation of the three networks.

7.1 Performance

Since our target is to model high quality 3D hairstyles with 2D sketches, the performance of our *DeepSketchHair* system is recorded in the following two aspects.

Qualitative Results. We have evaluated the performance of our *DeepSketchHair* system on some Internet portrait images and cartoon images, for which it is challenging for frontier image-based hair modeling methods to deal with. Note that, our system can generate 3D hair models from portrait images automatically by first using the method in [29] to generate hair mask images and dense 2D orientation maps. Meanwhile, users are allowed to perform freeform edits atop the creation. The hairstyles in all these cases are of various shapes and details. Some of the results are shown in Fig. 11, and 12. In Fig. 11, some of the examples are not

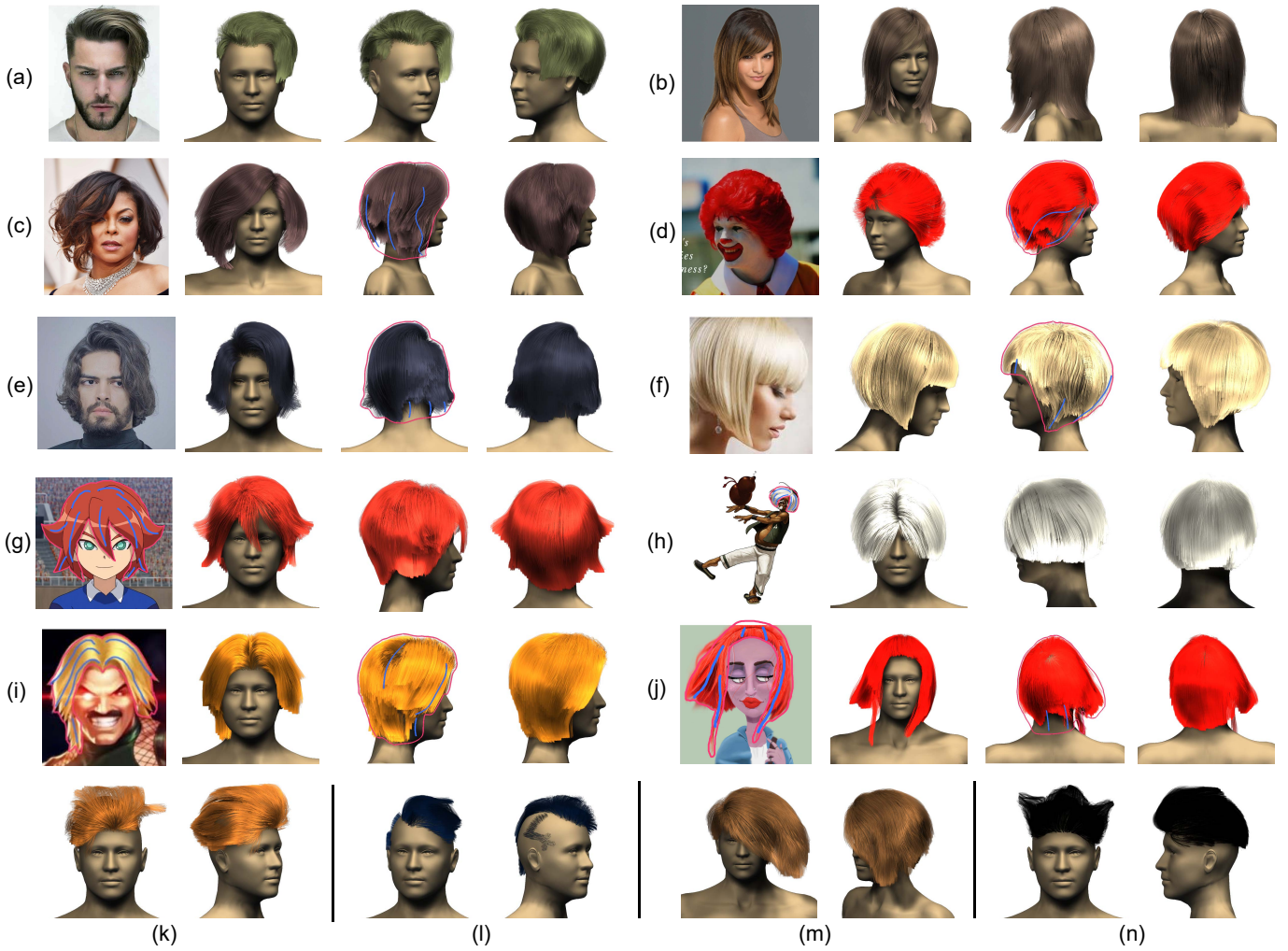


Fig. 11: A gallery of modeling results of 3D hairstyles from different inputs. Hairstyles in (a) and (b) are modeled from single images automatically. From left to right in (c) to (j), we show the reference images, the generated results, results in a different view, and the results after further edits by multi-view modeling (for g and h, no further edits were performed). The hairstyles created from scratch are shown in (k), (l), (m) and (n). The input sketches if any are overlaid with the reference images.

taken in frontal view (e.g., (d), (f), and (h)), some of them are of low resolution (e.g., (h)) while some of them are in cartoon format in which the hairstyles are dramatically different from real hairs ((g) - (j)). All aforementioned issues pose potential challenges for existing single-view image-based techniques as they either require frontal view or high-resolution input images, or rely on existing hair databases [8], [9]. The 3D hair models in the first row of Fig. 11 are generated automatically while the 3D hairstyles in the second to fifth rows are obtained after inputting additional sketches in new views to get more vivid results. The last row shows examples of freeform hairstyles that are created from scratch. Each hairstyle took less than 2 minutes to model by a trained user. The statistics including the number of views and the number of strokes to produce the associated results are shown in Table 1. Note our single-view modeling does not require a frontal view to start (e.g., Fig. 11, f) since we have training data from distant views (§4.3).

We have the following findings from the qualitative results: 1) Our *DeepSketchHair* system can generate state-of-the-art image-based hairstyles; 2) The generated results are highly realistic, no matter if the reference images are unreal cartoon pictures or real portrait images. 3) Our system is capable of generating various crazy hairstyles such as those in the last row of Fig. 11. All of these demonstrate the effectiveness and expressiveness of our system.

Robustness to the input sketches and view changes. Fig. 13 shows that our method is insensitive to varying input sketches depicting similar hairstyles. Even with random, broken, inconsistent, or very sparse sketch inputs, our method can consistently generate the desired high-quality results. In Fig. 14, we show that our *V2VNet* is robust against view changes, thanks to the randomly synthesized viewpoints in our training dataset.

Pilot Study. We invited five novice users with no 3D modeling experience and no training with our system. We first let them get familiar with our user interface and the functions of our system. Then we supplied them with some reference Internet images and taught them how to model a hairstyle from one of images using our system. The training step took about 5 minutes for each participant. In the evaluation test, the users were asked to do hair modeling tasks with a set of randomly selected images (3 for each) and try to do some artistic design. The sketches were created by using a Wacom touch screen with digital pen interface. We show some representative generated results and the corresponding reference images with the input sketches overlaid in Fig. 12. The

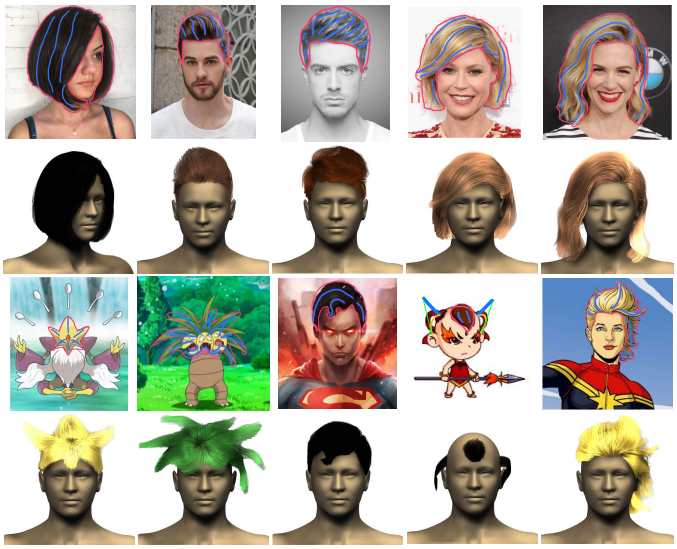


Fig. 12: Representative 3D hairstyles created by 5 novice participants (reference images and the corresponding sketches are shown atop).

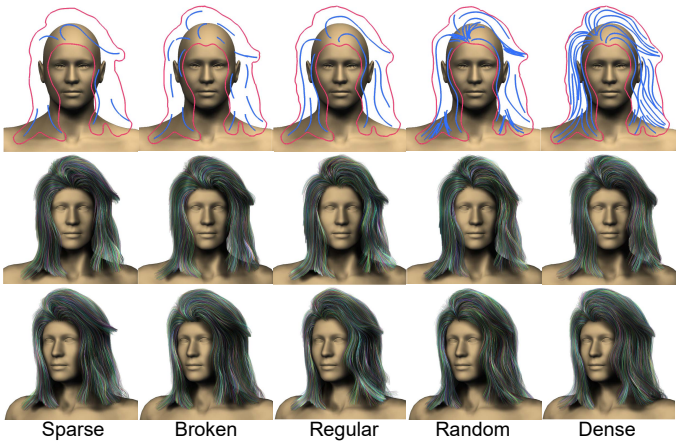


Fig. 13: Robustness of our method against varying input sketches depicting similar hairstyles.

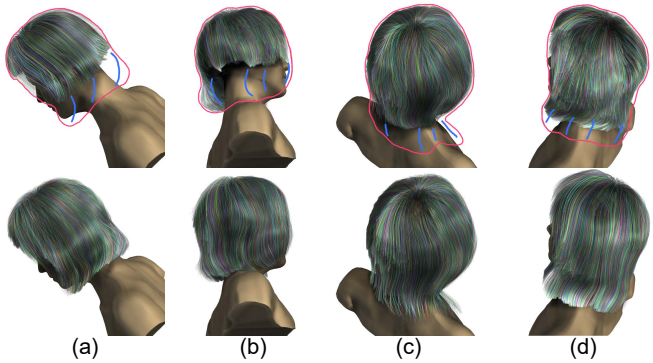


Fig. 14: Hair adjustments in random views. The first row shows sketches on top of initially synthesized hair models and the second row shows corresponding results.

Step	User Drawing	*S2ONet*	*O2VNet*	*V2VNet*	Hair Synthesis
Time	≈ 30s	≈ 0.2s	≈ 0.3s	≈ 0.2s	≈ 5s

TABLE 2: Average performance of our various algorithmic components in the pilot study.

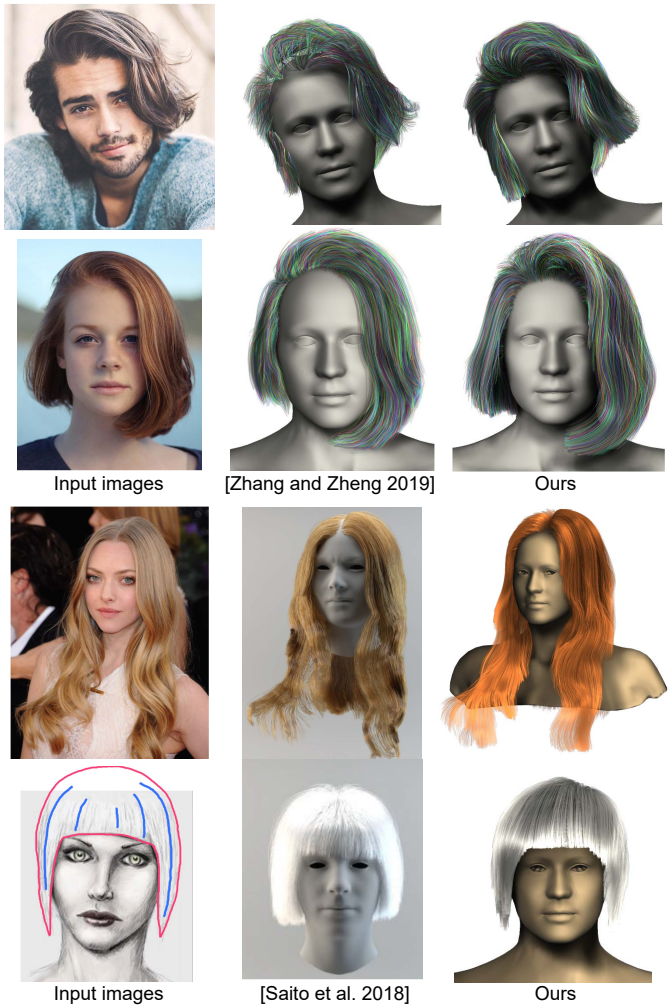


Fig. 15: Comparisons of our method with single-view image-based hair modeling methods.

average timing statistics are reported in Table 2. The users could get a generated 3D hair model in a few seconds when they finished their drawing and clicked the "build" button. Feedbacks from the participants were mainly positive but with the main concern that the response time of a few seconds is a bit long. Some of them expressed that the sketching interface in 2D is somehow limited when they wanted to deform and manipulate strands in 3D. Yet they all expressed our *DeepSketchHair* system can produce user-wanted hair models by simple interaction.

7.2 Comparisons

To the best of our knowledge, our method is the first method that can model high quality 3D hair models from sets of multi-view sketches. We compare our method with the state-of-the-art single-view image-based hair modeling methods [8], [9] and some sketch-based methods [5], [15], [17].

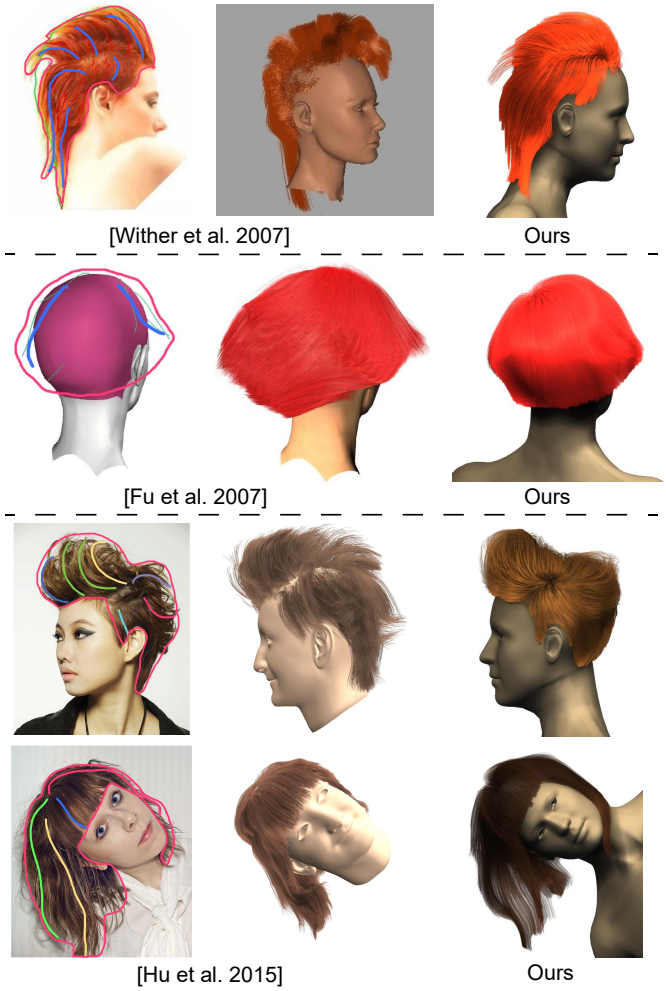


Fig. 16: Comparisons of our method with previous sketch-based hair modeling techniques. In the last two rows, we use the same set of strokes as in [5].

We have compared our *O2VNet* with *HairGAN* proposed by Zhang and Zheng [9]. For fair comparison, we generate the hair mask and dense 2D orientation map from the input image as in [9] and feed them into both HairGAN and our *O2VNet* (note that user sketches are not used here). Our generated hair strands have higher quality even without being post-processed (in Zhang and Zheng [9], tedious post-processing is required for refinement). As shown in the first two rows of Fig. 15, our hair strands are smoother and match the input images better.

The method by Saito et al. [8] can produce 3D hair strands from cartoon images automatically. With simple or no interaction (shown in Fig. 15), our *DeepSketchHair* can also generate 3D hair strands referring to cartoon images. Compared to [8], our results can fit cartoon hairstyles better with a small amount of user interaction.

Earlier sketch-based hair modeling methods [15] rely on the physically-based hair wisp model, which cannot maintain the balance between reality and user input sketches. As the first row of Fig. 16 shows, the result of [15] sometimes does not match the input strokes well, while our result is highly realistic and respects the input sketch well. We also compare our method with [17], which requires user-specified 3D curves to constrain the 3D orientation field by solving a Laplacian system. In the second row of Fig. 16, the readers can find that our result looks more reasonable, while the hair strands in their result are twisted in some degree. What's more, our method, which uses neural network forward propagation, is much faster than solving a large sparse linear system [17].

The recent data-driven sketch-based hair modeling approach by Hu et al. [5] relies on the reference images and dataset-based retrieval. This method combines different retrieved hairstyles to generate the most similar 3D hair model to the reference images. Thus, their generated hairstyles appear mixed and they cannot handle well when a target hairstyle matches none of those in the database like the examples shown in the last two rows of Fig. 16. Our learning-based method does not have these issues. In addition, different from [5], our method allows users to modify hairstyles from different views, so finally, we can generate 3D hair models with very complete structures. For fair comparisons, our results shown in Fig. 14 do not use additional sketches in new views.

7.3 Ablation Studies

In this subsection, we introduce the ablation studies to show the impact of some important algorithmic components in our system.

S2ONet. To test the impact of our *S2ONet*, we have explored three alternative structures:

(i) Take the same input of *S2ONet*, we simply diffuse the stroke direction to fill the entire hair mask region by solving a Laplace equation.

(ii) We use randomly selected 2D hair strands as the input of training without 2D sketch map pre-processing mentioned in §4.3. This experiment shares the same network structure with *S2ONet* and is trained for 100 epochs, too.

The average test errors are reported in Table 3, where all the methods are compared against the same ground truth. We find that our full *S2ONet* achieves the best performance.

(iii) We train our *O2VNet* with 2D sketches, 2D mask images, and bust depth images directly to examine the importance of the intermediate dense 2D orientation map (i.e., not using *S2ONet* at all). We call this net *S2VNet*. The evaluation results are shown in Fig. 17 (Left). It can be seen that without dense 2D feature maps, the details of generated 3D orientation fields are far away from the ground truth, which is consistent with our inference in §4.

O2VNet. We conducted the following experiments to evaluate our *O2VNet*.

(i) We first compared our network structure with *HairGAN* [9]. In this experiment, we train these two networks with the same dataset and loss function until they converged. The 2^{nd} and 3^{rd} columns of Fig. 17 show the differences between two network outputs. The inputs to the two networks are both derived from 3D hair models in the test set. The readers can find that not only the errors of our network outputs are lower, but also our synthesized 3D orientation fields are more similar to the ground truth with higher smoothness. Further, our network converges faster than *HairGAN* due to our much smaller input size.

Methods	Mean Square Error (MSE)
Laplacian Diffuse	6.98
Random Selection	6.60
S2ONet	**4.01**

TABLE 3: Ablations study on *S2ONet*.

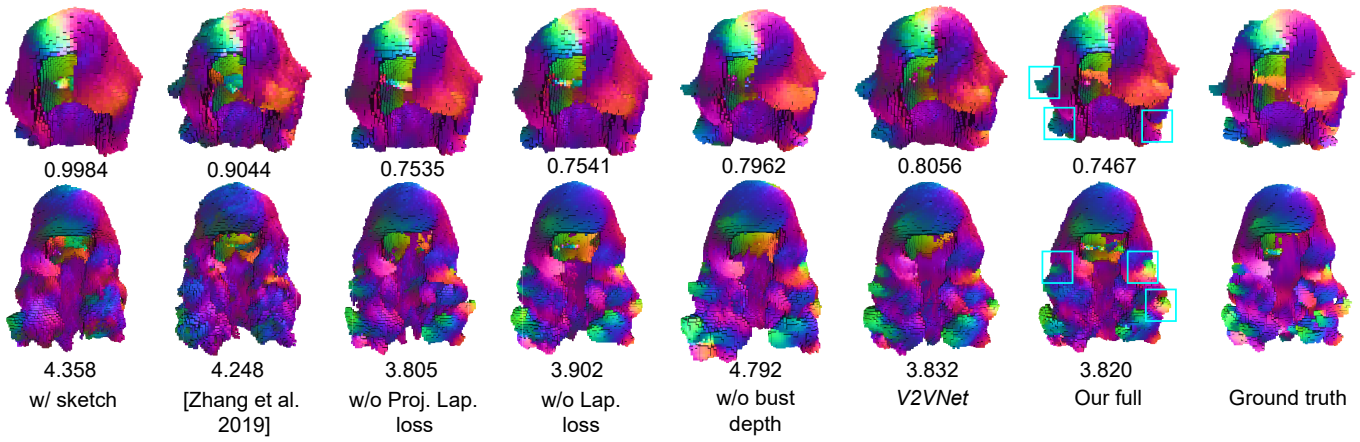


Fig. 17: Ablations study on *O2VNet*. The projection and Laplacian losses help our results generalize well to the inputs. The numbers indicate the MSE errors.

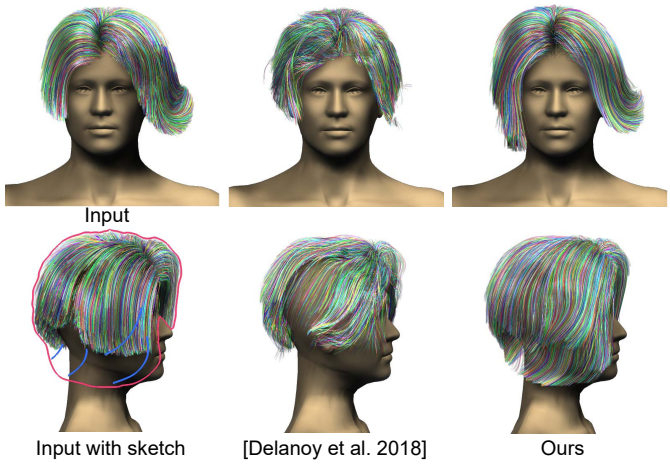


Fig. 18: Ablations study on *V2VNet*. The method by Delanoy et al. [38] tends to generate vector fields whose values are inconsistent with the ones in the previous view and thus are often chaotic (the middle column). For each method, we show the corresponding results in the previous and current views.

(ii) We also examined the effects of the projection and Laplacian losses. As shown in Fig. 17, although quantitatively the errors without both the projection loss and the Laplacian loss (the 3^{rd} column) are comparable to those with our full model, one can see visually that the results with the projection and Laplacian losses have higher curling degree, better matching with the ground truth. Besides, the results without the Laplacian loss (the 4^{th} column) get slight improvements over the details, but present more global errors compared with our full model.

(iii) The 5^{th} column shows that training our network without the depth as guidance results in a degradation. We also conducted an experiment to replace our *O2VNet* with *V2VNet*. In this regard, we take the voxelized bust model and the 2D orientation map as input to our *V2VNet* to generate a 3D vector field. However, we find a degeneration in the performance as shown in the 6^{th} column. This is partially due to that the lack of orientation guidance as input would prevent the 3D convolutional structure in the *V2VNet* to go deeper.

Volume-to-volume Generation. Although Delanoy et al. [38] present an update CNN dealing with multi-view sketch inputs to modify 3D surface models, we find this approach does not work well in our multi-view hair modeling task. We did an experiment to compare our volume-to-volume network structure with their method, which concatenates 3D volume with 2D feature maps and feeds them to a network using 2D convolutional layers only. In this experiment, we also train these two networks with the same dataset and the same loss function (we do not add the projection and laplacian loss). The readers can notice from Fig. 18 that our network performs much better than the update CNN in [38]: our method well preserves the features in the previous view and fills the user-specified empty mask region well.

7.4 Limitations

Our method has several limitations. There still exist complicated hairstyles that our *DeepSketchHair* system cannot deal with (Fig. 19). For example, some hairstyles like Afro (Fig. 19 (Bottom)) might have fine hair details which are difficult to handle with our pre-set resolution of 3D orientation fields. Similar lack-of-fine-detail effects can also be observed from Fig. 11 (c) and (d). This is a common drawback of the current deep learning based hair

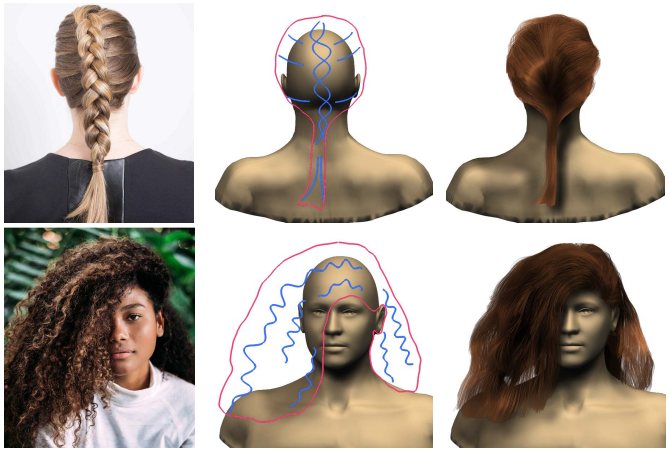


Fig. 19: Our method might fail to synthesize hair details due to either insufficient resolution of 2D or 3D orientation fields, or insufficient training data.

modeling approaches that rely on a vector field representation [8], [9]. Achieving higher resolutions will bring in tremendous computational cost and need GPUs with bigger memory for training. In addition, as a learning-based approach, the effectiveness of our method is also determined by the available training data. Since we have no hairstyles like polystrip in our dataset, it is difficult for our network to learn such special shapes (Fig. 19 (Top)).

8 Conclusions

In this work, we have presented a novel sketching system *DeepSketchHair* for hair modeling. We showed that a deep learning based method established on a 3D hairstyle dataset is able to generate various realistic 3D hair models in accordance with 2D sketches, in single or multiple views. With a simple sketch or an image, users can get a corresponding generated 3D hair model and then optionally perform further manipulations to get desired results. Our system is composed of three deep learning modules which are carefully designed to support seamless user interactions. The experimental results show the outstanding performance of our modeling system, and demonstrate the effectiveness and expressiveness of our proposed method. In our current implementation, we require users to draw the hair mask in full (even for editing in new views) and the directed strokes to depict the hair growing direction. More intelligent interfaces or alternative interactions (for examples, those in VR) could be investigated to ease the interactions and to enable more powerful hair editing in 3D.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This work was supported in part by the National Key Research & Development Program of China (2018YFE0100900), NSF China (No. U1609215), and the Fundamental Research Funds for the Central Universities. Hongbo Fu was supported by a gift from Adobe and grants from the RGC of HKSAR (Project No. CityU 11212119), City University of Hong Kong (Project No. 7005176), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU.

References

[1] M. Zhang, M. Chai, H. Wu, H. Yang, and K. Zhou, "A data-driven approach to four-view image-based hair modeling," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 156:1–156:11, 2017.

[2] L. Hu, S. Saito, L. Wei, K. Nagano, J. Seo, J. Fursund, I. Sadeghi, C. Sun, Y.-C. Chen, and H. Li, "Avatar digitization from a single image for real-time rendering," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 195:1–195:14, 2017.

[3] M. Chai, L. Wang, Y. Weng, Y. Yu, B. Guo, and K. Zhou, "Single-view hair modeling for portrait manipulation," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 116:1–116:8, 2012.

[4] M. Chai, L. Wang, Y. Weng, X. Jin, and K. Zhou, "Dynamic hair manipulation in images and videos," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 75:1–75:8, 2013.

[5] L. Hu, C. Ma, L. Luo, and H. Li, "Single-view hair modeling using a hairstyle database," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 125:1–125:9, 2015.

[6] M. Zhang, P. Wu, H. Wu, Y. Weng, Y. Zheng, and K. Zhou, "Modeling hair from an rgb-d camera," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 205:1–205:10, 2018.

[7] Y. Zhou, L. Hu, J. Xing, W. Chen, H.-W. Kung, X. Tong, and H. Li, "Hairnet: Single-view hair reconstruction using convolutional neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 235–251.

[8] S. Saito, L. Hu, C. Ma, H. Ibayashi, L. Luo, and H. Li, "3d hair synthesis using volumetric variational autoencoders," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 208:1–208:12, 2018.

[9] M. Zhang and Y. Zheng, "Hair-gan: Recovering 3d hair structure from a single image using generative adversarial networks," *Visual Informatics 2019*, vol. 3, no. 2, pp. 102–112, 2019.

[10] Y. Watanabe and Y. Suenaga, "A trigonal prism-based method for hair image generation," *IEEE Computer Graphics and applications*, vol. 12, no. 1, pp. 47–53, 1992.

[11] T.-Y. Kim and U. Neumann, "Interactive multiresolution hair modeling and editing," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 620–629, 2002.

[12] B. Choe and H.-S. Ko, "A statistical wisp model and pseudophysical approaches for interactive hairstyle generation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 2, pp. 160–170, 2005.

[13] X. Mao, S. Isobe, K.-i. Anjyo, and A. Imamiya, "Sketchy hairstyles," in *International 2005 Computer Graphics*, 2005, pp. 142–147.

[14] S. Malik, "A sketching interface for modeling and editing hairstyles," in *SBM*, 2005, pp. 185–194.

[15] J. Wither, F. Bertails, and M.-P. Cani, "Realistic hair from a sketch," in *IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07)*, 2007, pp. 33–42.

[16] Y. Yu, "Modeling realistic virtual hairstyles," in *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001*, 2001, pp. 295–304.

[17] H. Fu, Y. Wei, C.-L. Tai, and L. Quan, "Sketching hairstyles," in *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, 2007, pp. 31–36.

[18] J. Xing, K. Nagano, W. Chen, H. Xu, L.-y. Wei, Y. Zhao, J. Lu, B. Kim, and H. Li, "Hairbrush for immersive data-driven hair modeling," in *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology*. ACM, 2019, pp. 263–279.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[20] K. Ward, F. Bertails, T.-Y. Kim, S. R. Marschner, M.-P. Cani, and M. C. Lin, "A survey on hair modeling: Styling, simulation, and rendering," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 2, pp. 213–234, 2007.

[21] Y. Bao and Y. Qi, "A survey of image-based techniques for hair modeling," *IEEE Access*, vol. 6, pp. 18 670–18 684, 2018.

[22] L.-H. Chen, S. Saeyor, H. Dohi, and M. Ishizuka, "A system of 3d hair style synthesis based on the wisp model," *The Visual Computer*, vol. 15, no. 4, pp. 159–170, 1999.

[23] X. D. Yang, Z. Xu, J. Yang, and T. Wang, "The cluster hair model," *Graphical Models*, vol. 62, no. 2, pp. 85–103, 2000.

[24] Z. Xu and X. D. Yang, "V-hairstudio: an interactive tool for hair design," *IEEE Computer Graphics and Applications*, vol. 21, no. 3, pp. 36–43, 2001.

[25] K. W. Ming, T. Hiroki, and N. Masayuki, "Generation of 3d hair model from 2d image using image processing," in *Applications of Digital Image Processing XIX*, vol. 2847, 1996, pp. 303–312.

[26] S. Paris, H. M. Briceño, and F. X. Sillion, "Capture of hair geometry from multiple images," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 712–719, 2004.

[27] Y. Wei, E. Ofek, L. Quan, and H.-Y. Shum, "Modeling hair from multiple views," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 816–820, 2005.

[28] M. Chai, L. Luo, K. Sunkavalli, N. Carr, S. Hadap, and K. Zhou, "High-quality hair modeling from a single portrait photo," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 204:1–204:10, 2015.

[29] M. Chai, T. Shao, H. Wu, Y. Weng, and K. Zhou, "Autohair: Fully automatic hair modeling from a single image," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 116:1–116:12, 2016.

[30] P. K. Diederik, M. Welling *et al.*, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[31] C. Li, H. Pan, Y. Liu, X. Tong, A. Sheffer, and W. Wang, "Robust flow-guided neural prediction for sketch-based freeform surface modeling," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 238:1–238:12, 2018.

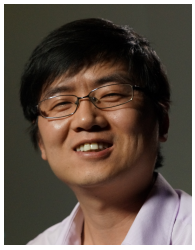
[32] T. Igarashi, T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 409–416.

[33] Y. Mori and T. Igarashi, "Plushie: An interactive design system for plush toys," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

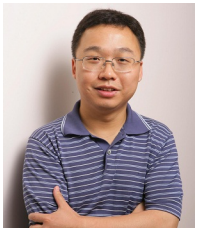
[34] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Fibermesh: Designing freeform surfaces with 3d curves," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.
[35] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3-sweep: Extracting editable objects from a single photo," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 195:1–195:10, 2013.
[36] A. Shtof, A. Agathos, Y. Gingold, A. Shamir, and D. Cohen-Or, "Geosemantic snapping for sketch-based modeling," in *Computer graphics forum*, vol. 32, no. 2pt2, 2013, pp. 245–253.
[37] H. Huang, E. Kalogerakis, E. Yumer, and R. Mech, "Shape synthesis from sketches via procedural models and convolutional networks," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 8, pp. 2003–2013, 2016.
[38] J. Delanoy, M. Aubry, P. Isola, A. A. Efros, and A. Bousseau, "3d sketching using multi-view deep volumetric prediction," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 1, no. 1, pp. 21:1–21:22, 2018.
[39] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji, and R. Wang, "3d shape reconstruction from sketches via multi-view convolutional networks," in *2017 International Conference on 3D Vision (3DV)*, 2017.
[40] W. Su, D. Du, X. Yang, S. Zhou, and H. Fu, "Interactive sketch-based normal map generation with deep neural networks," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 1, no. 1, pp. 22:1–22:17, 2018.
[41] X. Han, C. Gao, and Y. Yu, "Deepsketch2face: A deep learning based sketching system for 3d face and caricature modeling," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 126:1–126:12, 2017.
[42] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.
[43] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.
[44] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5400–5409.
[45] L. Zhang, C. Li, T.-T. Wong, Y. Ji, and C. Liu, "Two-stage sketch colorization," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 261:1–261:14, 2018.
[46] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.
[47] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.
[48] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 234–241.
[49] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," *CoRR*, vol. abs/1704.06254, 2017.
[50] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1511–1520.
[51] L. Wang, Y. Yu, K. Zhou, and B. Guo, "Example-based hair geometry synthesis," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 56:1–56:9, 2009.
[52] A. Kar, C. Häne, and J. Malik, "Learning a multi-view stereo machine," in *Advances in neural information processing systems*, 2017, pp. 365–376.
[53] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "Deepvoxels: Learning persistent 3d feature embeddings," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2437–2446.
[54] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.
[55] G. Moon, J. Yong Chang, and K. Mu Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5079–5088.
[56] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 43:1–43:10, 2014.
[57] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, ser. SGP '04, 2004, pp. 175–184.

Yuefan Shen is a Ph.D. candidate at the State Key Lab of CAD&CG, Zhejiang University. He obtained his B.S. from the School of Software Engineering at Shandong University. His research interests include image-based modeling and 3D data processing with deep learning.

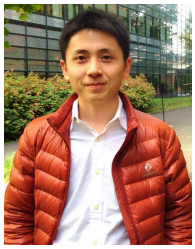
Changgeng Zhang is a Master student at the State Key Lab of CAD&CG, Zhejiang University. He obtained his B.S. from the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include image-based modeling and 3D data processing with deep learning.



Hongbo Fu received a BS degree in information sciences from Peking University, China, in 2002 and a PhD degree in computer science from the Hong Kong University of Science and Technology in 2007. He is a Full Professor at the School of Creative Media, City University of Hong Kong. His primary research interests fall in the fields of computer graphics and human computer interaction. He has served as an Associate Editor of The Visual Computer, Computers & Graphics, and Computer Graphics Forum.



Kun Zhou is a Cheung Kong Professor in the Computer Science Department of Zhejiang University. He received his B.S. degree and Ph.D. degree in computer science from Zhejiang University in 1997 and 2002, respectively. His research interests are in visual computing, parallel computing, human computer interaction, and virtual reality. He currently serves on the editorial advisory boards of ACM Transactions on Graphics and IEEE Spectrum. He is a Fellow of IEEE.



Youyi Zheng is a Researcher at the State Key Lab of CAD&CG, Zhejiang University. He received a BS degree and an MS degree in Mathematics, both from Zhejiang University, China, in 2005 and 2007, and a PhD in Computer Science from the Hong Kong University of Science & Technology in 2011. His research interests include geometric modeling, imaging, and human-computer interaction. He has served as an Associate Editor of The Visual Computer and Frontiers of Computer Science.