Towards Dynamic Hair Reconstruction: Reproduce of HairNet

Shiyang Jia
Shanghai Jiao Tong University
sher_locked@sjtu.edu.cn

Abstract

In this project, we explored the data-driven solution to realistic hair modeling problem. There is already some work on generating static hair model, therefore we want to improve on it and add some temporal and spatial coherence. In a word, our work is mainly to reproduce a static hair model generation algorithm based on CNN, and put forward the idea of introducing temporal and spatial coherence through RNN.

1. Introduction

Realistic hair modeling is one of the most difficult tasks when digitizing virtual humans. In contrast to objects that are easily parameterizable, like the human face, hair spans a wide range of shape variations and can be highly complex due to its volumetric structure and level of deformability in each strand. The problem becomes more complicated when it comes to reconstructing dynamic hair model, which requires to ensure temporal and spatial coherence.

However, the amount of articles relating to this problem is relatively small, especially for dynamic hair modeling. Recent literature mainly focus on the generation of static hair model, and amount of data-driven methods are applied. For instance, Chai *et al*. [1] adopted a convolutional neural network to segment the hair in the input image to fully automate the modeling process. Zhang *et al*. [7] learned an architecture of generative adversarial networks to recover the 3D hair structure from a single image. There are also some methods take multi-view images [6] or videos [3] as network input. On the other hand, dynamic hair modeling methods still stay in a traditional stage. Xu *et al*. [5] propose a motion-path analysis algorithm to track local hair motions in input videos, and formulate the global hair reconstruction as a spacetime optimization problem.

Based on the literature reviewing, we decided to use convolutional neural network adapted from HairNet [9] to generate original static hair model. HairNet is a relatively simple network without too many parameters, which means it is easier to train and adapt. Moreover, HairNet products a strand feature layer as medial output, this layer can be extracted and put into RNN to ensure the temporal and spatical coherence.

Our contributions in this project can by summarized as follows:

- We construct a small database of around 300 3D hair models and 1800 corresponding orientation maps.
- We achieve an effective representation of hair roots by introducing scalp-space. Then, sampling hair strand based on the representation.
- We put forward the idea of ensuring temporal and spatial coherence through RNN.

2. Approach Overview

In this section, data generation is introduced first. This step produces orientation maps corresponding to each synthetic hair model. Then, I will explain the representation methods we used to parameterize hairs. Finally, CNN takes the orientation maps as input and generates hair strands represented as sequences of 3D points.

2.1. Data Preprocessing

Obtaining a training set with real hair images and ground-truth 3D hair geometries is challenging. We can factor out the difference between synthetic and real hair data by using an intermediate 2D orientation field as network input. This enables our network to be trained with largely accessible synthetic hair models and also real images without any changes. For example, the 2D orientation field can be calculated from a real image by applying a Gabor filter on the hair region automatically segmented using the method of [8]. It can also by generated easily from synthetic 3D hair model by projection.

Orientation Map Generation. We collect an original hair dataset with about 300 3D artificial hair models provided by USC-HairSalon[1], which have already been aligned to an identical bust model. Then, we define a bounding box as the boundary of our model space. The center point of the

[1] http://www-scf.usc.edu/~liwenhu/SHM/database.html

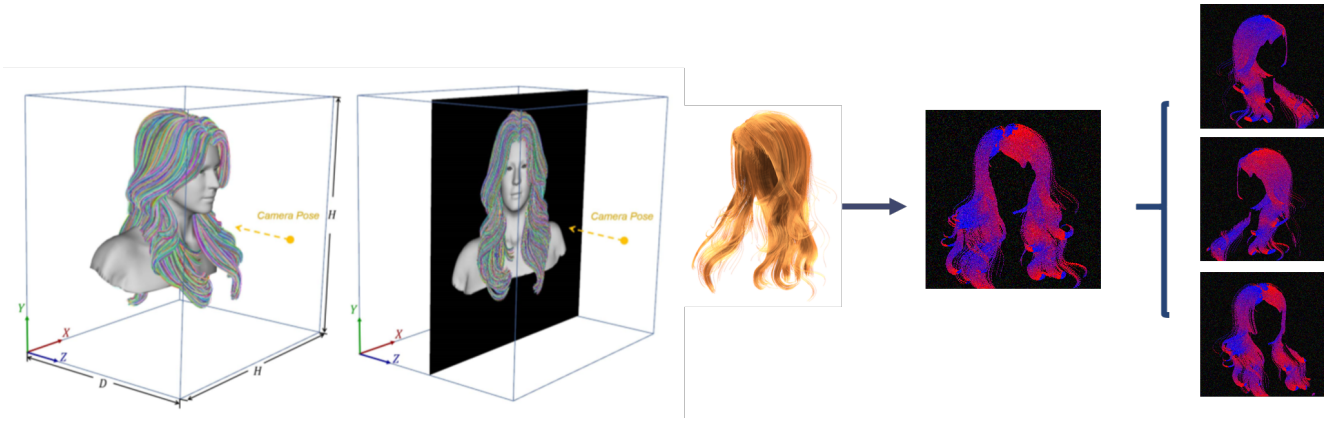

Figure 1. Left: A bounding box set up with a camera pose for 3D volume definition and 2D projection [7]. Right: Data augmentation by rotating and flipping.

bounding box is determined by the mean point among all hairs. In order to get 2D orientation map, we put a camera straight forward to the bust model. The center of the image plane coincides with the center of the bounding box. The 2D map is produces by orthogonal projection of strand's tangent direction. The final output is a $2 \times 256 \times 256$ image, whose two channels store the color-coded hair orientation map (see the left part of Figure 1).

Data Augmentation. We rotate the 3D model around the y-axis at random angles and randomly flip the generated images (see the right part of Figure 1). This operation can simulate the situation where human face is not directly facing the camera in real photos. In addiction, we also add Gaussian noises to the orientation to emulate the real conditions.

After this step, we obtain more than 2000 orientation map and corresponding hair models from about 300 original data.

2.2. Hair Model Representation

In order to make learning efficient, Hair model needs to be parameterized into an easy-to-learn form. Firstly, all of the hair strands need to be represented in a uniform form. Secondly, as a hairstyle contains 10K strands, hair sampling must by performed to speed up training.

Hair Strand Representation. We represent each strand as an ordered 3D point set $\zeta = \{s_i\}_{i=0}^{M}$, evenly sampled with a fixed number ($M = 100$ in our experiments) of points from the roots to end. Each sample s_i contains attributes of position $\mathbf{p}_i$ and curvature c_i (The curvature parameter is further used in reconstruction in original paper, but there we just treat it as an additional information feed into network). Although the strands have large variance in length, curliness, and shape, they all grow from fixed roots to flexible ends. To remove the variance caused by root positions, we represent each strand in the local coordinate anchored at its root.

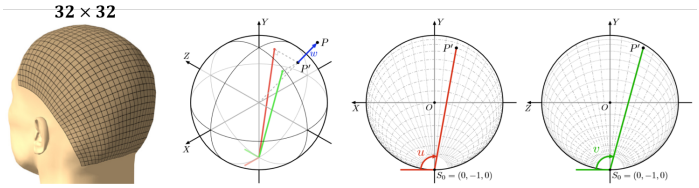

Figure 2. 3D scalp space parameterization [4]

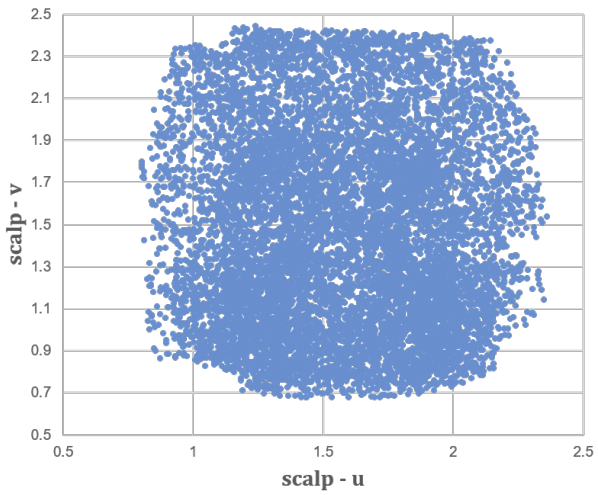

Figure 3. Hair root distribution in the uv plane of scalp space

Hair Sampling. We sample $N = 1024$ strands per hair model as training ground truth. In order to make uniform sampling among the scalp, we adopt the method in [4] to parameterize the scalp to a 32×32 grid, and sample hair roots at those grid centers.

Specifically, this method assume that the scalp surface is similar to the upper half of a unit sphere. The world coordinate frame is originated at the center of the hemisphere. Its y-axis points upward. Given a world space point

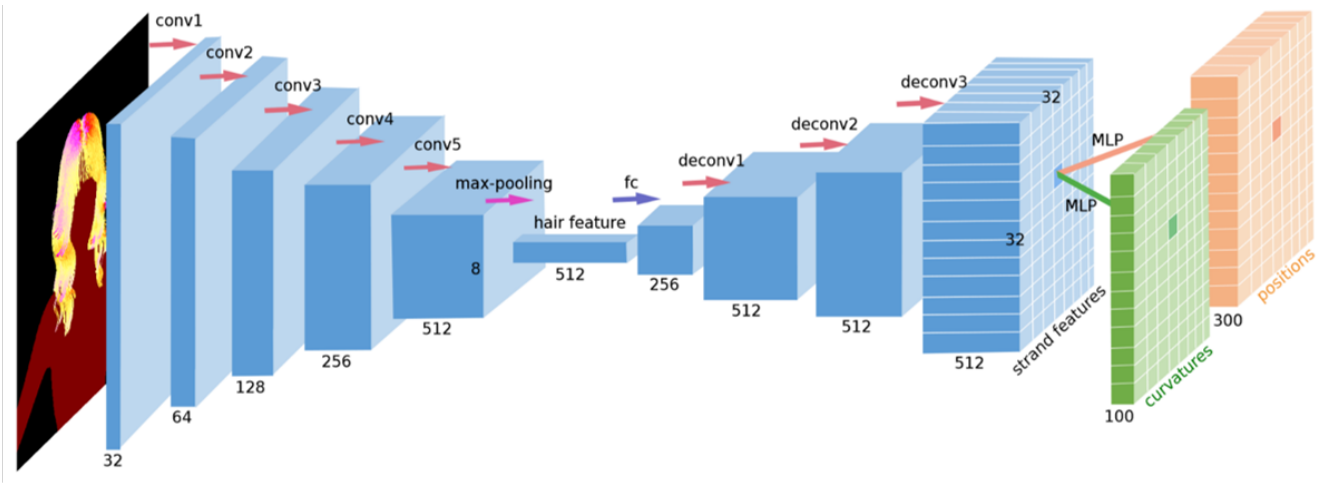

Figure 4. Network Architecture [9]

$P = (x, y, z)$, its scalp space coordinates (u, v) are defined as follows:

$$u = \arccos \frac{\hat{x}}{\sqrt{\hat{x}^2 + (\hat{y}+1)^2}}$$
$$v = \arccos \frac{\hat{z}}{\sqrt{\hat{z}^2 + (\hat{y}+1)^2}} \quad (1)$$

where $(\hat{x}, \hat{y}, \hat{z})$ is the spherical projection of (x, y, z) onto the unit sphere (see Figure 2). Intuitively, u and v can be interpreted as two angles ranging from 0 to π as shown in the two rightmost figures in Figure 2.

Hair root distribution in the uv plane of scalp space is shown in Figure 3. We can see that the scalp representation can align the root distribution with the coordinate axis so that uniform sampling can be performed.

Finally, the hair model can be treated as a set of N strands $H = \zeta^N$ with fixed roots, and can be formulated as a matrix $A_{N\times M}$, where each entry $A_{i,j} = (\mathbf{p}_{i,j}, c_{i,j})$ represents the jth sample point on the ith stand.

2.3. Hair Prediction Network

Network Architecture. As illustrated in Figure 4, our network first encodes the input image to a latent vector, followed by decoding the target hair strands from the vector. For the encoder, we use the convolutional layers to extract the high-level features of the image. We use the 2D max-pooling to spatially aggregate the partial features (a total number of 8×8) into a global feature vector z. This greatly reduces the number of network parameters.

The decoder generates the hair strands in two steps. The hair feature vector z is first decoded into multiple strand feature vectors $\{z_i\}_{i=0}^{M}$ via deconvolutional layers, and each z_i could be further decoded into the final strand geometry ζ via the same multi-layer fully connected network. This multi-scale decoding mechanism allows us to efficiently produce denser hair models by interpolating the strand features.

Loss Functions. We apply two losses on our network. They are the L_2 reconstruction loss of the 3D position and the curvature of each sample (We do not adopt the collision loss in the original paper because it is hard to fit human body with ellipsoids and the performance increase is slight):

$$L = \alpha_1 L_{pos} + \alpha_2 L_{curv}. \quad (2)$$

L_{pos} and L_{curv} are the loss of the 3D positions and the curvatures respectively, written as :

$$L_{pos} = \frac{1}{NM} \sum_{i,j} w_{ij} \|\mathbf{p}_{ij} - \mathbf{p}^*_{ij}\|_2^2$$
$$L_{curv} = \frac{1}{NM} \sum_{i,j} w_{ij} (c_{ij} - c^*_{ij})^2 \quad (3)$$

where $\mathbf{p}^*_{ij}$ and c^*_{ij} are the corresponding ground truth position and curvature to $\mathbf{p}_{ij}$ and c_{ij}.

Given a single-view image, the shape of the visible part of the hair is more reliable than the invisible part, *e.g.* the inner and back hair. Thus we assign adaptive weights to the samples based on their visibility visible samples will have higher weights than the invisible ones:

$$w_{ij} = \begin{cases} 10.0 & s_{ij} \text{ is visible} \\ 0.1 & \text{otherwise} \end{cases} \quad (4)$$

3. Experiment

In this section, I will briefly explain how we process hair model data and setup neraul network. Then, the network inference results are shown with some analysis.

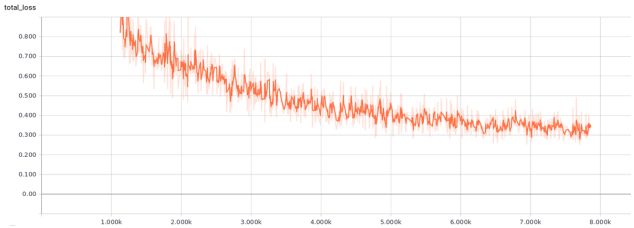

Figure 5. Total loss curve converges at around 7000K step

3.1. Implement Detials

We implement data preprocessing and hair sampling with C++. In this process, every hair model generates 6 orientation maps. The hair points' position and curvature are also calculated and stored in HDF5 file while performing projection. After that, we set 10% of these data as test set.

We use Tensorflow to construct and train CNN. During training, we resize all the hair so that the hair is measured in the metric system. Specifically, the training parameters of Equation 2 are fixed to be $\alpha_1 = \alpha_2 = 1$. We use Relu for nonlinear activation, Adam [2] for optimization, and run the training for 50 epochs using a batch size of 16 and learning rate of 10^{-4}.

3.2. Result Analysis

The total loss curve shown in Figure 5 seems to converge, but as we measure the mean square distance between all the points on test hair model, the error fluctuates between 6.4 cm and 7.0 cm. It is pretty large compared to 1.7 cm in the original paper. We randomly visualize a few predictions, the results are shown in Figure 6. We can see the network predictions can only represent outline of a hairstyle but ignore many strand details. Some hair strands in the back even tilt up from scalp, but this phenomenon gradually eases as the training step increase.

I think the main reason for the unsatisfactory result is that the size of dataset is too small. Every hair sample contains about 10^6 points, but we only use 1800 samples to train the network. So, it becomes a big p small n problem. Compared with 40K hair models used in the original paper, our network will suffer overfitting quickly while training with 1800 hair models. Additionally, visible-weight does not make sense in this inaccurate training process.

4. Conclusion & Future Work

In this project, we explored the idea of using neural networks to generate dynamic hair model. Due to limited time and device, we only model static hair without any open source code. This project made me realize that there are many details and difficulties behind a novel idea. This project involves not only the construction and training of

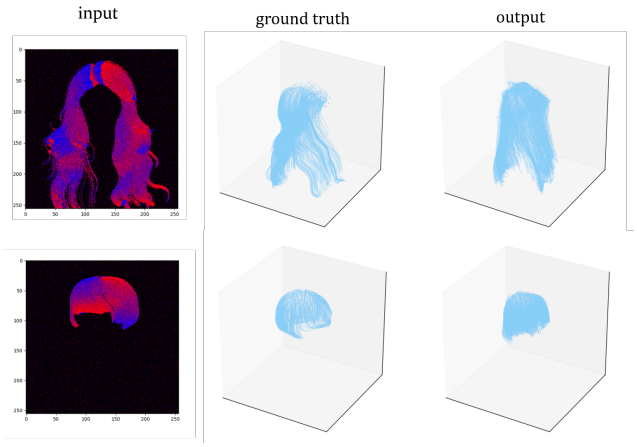

Figure 6. Visualized result

neural network, but also many graphical operations in 3D space. The time I spend on making dataset was even longer than training the network.

In the future, I will expend the size of dataset and refine the smoothness and curliness of the hair to get plausible result. What's more, I will try to take the video sequence as input, then refine the strands-feature layers through RNN to ensure temporal and spatial coherence. After interpolating among features and reconstruction, the dynamic hair model matching the input video will be generated.

[1] M. Chai, T. Shao, H. Wu, Y. Weng, and K. Zhou. Autohair: fully automatic hair modeling from a single image. *ACM Trans. Graph.*, 35:116:1–116:12, 2016.

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[3] S. Liang, X. Huang, X. Meng, K. Chen, L. G. Shapiro, and I. Kemelmacher-Shlizerman. Video to fully automatic 3d hair model. *ACM Trans. Graph.*, 37:206:1–206:14, 2018.

[4] L. Wang, Y. Yu, K. Zhou, and B. Guo. Example-based hair geometry synthesis. In *SIGGRAPH '09*, 2009.

[5] Z. Xu, H.-T. Wu, L. Wang, C. Zheng, X. Tong, and Y. Qi. Dynamic hair capture using spacetime optimization. *ACM Trans. Graph.*, 33:224:1–224:11, 2014.

[6] 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.*, 36:156:1–156:11, 2017.

[7] M. Zhang and Y. Zheng. Hair-gans: Recovering 3d hair structure from a single image. *CoRR*, abs/1811.06229, 2018.

[8] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, 2017.

[9] 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 *ECCV*, 2018.