3D Hair Reconstruction based on Hairstyle Attributes Learning from Single-view Hair Image using Deep Learning

by

Chao Sun

Thesis submitted to the University of Ottawa

in partial fulfillment of the requirements for the

Doctorate of Philosophy degree in

Electrical and Computer Engineering

uOttawa

School of Electrical Engineering and Computer Science

Faculty of Engineering, University of Ottawa

Ottawa, Ontario

Abstract

Hair, as a vital component of the human's appearance, plays an important role in producing digital characters. However, the generation of realistic hairstyles usually needs professional digital artists and/or complex hardware, and the procedure is often time-consuming due to its numerous numbers, and diverse hairstyles. Thus, automatic capture of real-world hairstyles with easy input can greatly benefit the production pipeline.

State-of-the-art 3D hair modeling systems require either multi-view images or a single-view image with complementary synthetic 3D hair models. For the multi-view image based 3D hair reconstruction, the capture systems are often made of a large number of cameras, projectors, light sources, and are usually in the indoor environment, which prevents popular use of the methods. On the contrary, single-view image based methods only use simple capture devices, eg; a handheld camera. However, a front view containing a face is often required and the resulting 3D hair strand reconstruction quality is compromised. Meanwhile, several hairstyles can not be easily modeled, such as braids and kinky hairstyles (afro-textured hairs), even though they are very common in real life.

In this dissertation, we implement a single-view imaged based 3D hair modeling system, where our hair reconstruction is done through 2D hair analysis and 3D strands creation, which benefits from both traditional image processing techniques and the strengths of machine learning. Our 2D hair analysis is used to learn the attributes of input hairs, including 2D hair strands, detailed hairstyle patterns, and the corresponding parametric

representation (which includes braids and kinky hairs), and braid structures. Simultaneously 3D hair strands are generated using deep-learning models. Our method is different from previous methods as our generated hair models can be modified by controlling the attributes and parameters we learned from the 2D hair recognition/analysis system.

Our system does not require a face to be shown in the input image and to our best knowledge, our work is the first work that can reconstruct 3D braided hair and kinky hair given a single-view image. Qualitatively and quantitatively assessments indicate that our system can generate a variety of realistic 3D hairstyle models.

Acknowledgements

I would like to express my sincere appreciation to Dr. Won-Sook Lee for her constant support during my doctorate studies. Her critical comments and insightful feedbacks were essential for any steps toward the progress of my research. Prof. Lee not only supports my study but also encourages me all the time in my life, especially during the special period of Covid-19. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I also would like to thank all my friends and lab members for their in-depth knowledge, support, and constructive feedback.

Dedication

To my mother, father, husband, and son. For their endless love, support, and encouragement.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Hair is a vital component of a person's identity, which can provide strong cues about age, background, and even personality [87]. Hair modeling is an important part of many computer graphic applications. For example, it can help to create a convincing virtual character in computer games and movies by providing a certain identity and personality

based on different hairstyles. Another example is in the cosmetic industry, 3D hair models with different hairstyles can be applied to the avatars of the customers to help the customers to design suitable hairstyles [73].

However, hair is also one of the most challenging objects to reconstruct. A human head typically consists of over 100,000 hair strands and each hair strand is quite small in diameter ranging from 17 μm to 181 μm [87]. Furthermore, hair usually has a complex structure and suffers from occlusions [61]. Besides, the color, brightness, and styles of hair always affect the difficulty of capturing, analyzing, and reconstructing.

Production-level 3D hair reconstruction systems typically require sophisticated capture systems (with many cameras and complex setups), controlled environments (indoor environment with controlled light sources), and manual work. Nowadays, in order to generate photo-realistic virtual characters for computer games and movies, 3D hair models are still created manually by highly trained and experienced digital artists, with the help of particular design tools (e.g. HairFarm, or Ornatrix). Those hair models have good quality and are animatable. Furthermore, creativity and imagination can be easily added to those models. However, the procedure is time-consuming. Even for a skilled user, compelling and realistic hairstyles can easily take days or weeks to produce [94]. Thus building a 3D hair reconstruction system for end-user needs the trade-off among the capture systems, robustness of the methods, and the quality of reconstructed results.

The goal of this study is to use the rich attributes of different hairstyles from single-view hair images to generate realistic 3D hair models of variety hairstyles. Thus our proposed system is a hierarchy work that ranges from 2D hair analysis and physical-plausible 3D hair strands reconstruction since the ultimate goal of 3D hair modeling is to obtain 3D hair strands that are realistic and can be easily animated.

1.2 Objectives

We want to build a single-view image-based 3D hair modeling system. The input is a single-view hair image without view constraints. Our hair modeling system can automatically generate 3D hair models of various hairstyles, including straight hairs, curly hairs, and two of the most challenging hairstyles- kinky hairs and braided hairs. The generated 3D hair models are strand-based which can be anchored to the scalp and be animated later.

Previous single-view hair modeling methods focus on recovering the 2D and 3D orientation fields to guide the growth of 3D hair strands. However, all those methods fail to reconstruct the 3D kinky hairs due to the noisy orientation fields. Furthermore, those methods can not correctly analysis the overlapping structure of braids. We aim to utilize 2D hair image analysis to provide useful information to generate realistic 3D hair models, especially for kinky hairs and braided hairs.

In order to achieve this goal, we need to: (1) extract the strand structure from a hair image, (2) recognize the hairstyle patterns and learn the parametric representation of different hairstyles, (3) discover the 2D/3D structure of braided hairstyles, (4) generate 3D hair models with reasonable geometry from one input image under no view constraints, (5) modify the 3D strands by the parameters that are learned from image and generate visually plausible 3D hair models in various hairstyles. We approach these tasks using image processing techniques, and deep-learning models and exploit existing datasets.

1.3 Challenges

State-of-the-art 3D hair model reconstruction methods require either multi-view images or a single-view image with complementary synthetic 3D hair models. Multi-view 3D hair reconstruction systems often need complex capture systems and manual work. On the contrary, single-view 3D hair modeling systems only require simple capture equipment, but the input image is restricted to a limited view range (which means that a face must be shown in the image).

In addition, current hair modeling systems can only generate straight or curly hairs and fail to reconstruct braided hairs and kinky hairs. On the contrary, our method is based on 2D hair image analysis and 3D hair strands generation. Both parts utilize the power of deep learning deep learning. Constructing deep-learning based systems requires data. However, collecting proper datasets for 2D hair analysis and 3D hair reconstruction is difficult due to the following reasons: (1) hair image analysis and 3D hair modeling are not very popular topics, thus researchers often create datasets from scratch, (2) plenty of 2D hair images can be downloaded from the Internet, but we don't have access to their 3D volumes nor 3D strands, (3) existing 3D hair model datasets provide 3D hair models but do not come with real 2D hair images, (4) existing 3D hair model datasets lack some complex and constrained hairstyle models, such as the braid hairstyle models, kinky hairstyle models, etc.

1.4 System Overview

The overview of the thesis work is shown in Figure 1.1. The input of the 3D hair modeling system is a single-view hair image that only has one person with hair in the image. The subject is relatively close to the camera and there is no view constraints on the object, it can be in the range from front-view to back-view.

Given the single-view hair image, we perform 2D hairstyle pattern recognition to recognize the hairstyles within the hair region. Then based on the hairstyle, we divided the hairs into two categories: unconstrained hairstyles (straight, curly, and kinky hair) and braided hairstyles. For unconstrained hairstyles, we generated the 3D guide hair strands using deep learning networks, then we also perform the hairstyle representation analysis to extract the parameters based on the hairstyles. Then we adjust the 3D hair guide strands by applying the hairstyle parameters to generate the final 3D hair models of various hairstyles, including very curly hairs and kinky hairs. For braided hairstyles, we further divided them into two cases. The first case is that the existence of braids doesn't change the shape of the non-braid region. In this situation, we generate the 3D hair strands in the non-braid region using our deep-learning network. Meanwhile, we recover the braid structure based on braid unit recognition. Then we pop the braids on top of the non-braid region to have the final braided hair models. In the second case, the braids change the shape of the non-braid region. We recover the 3D hair volume based on the hair region silhouette. We also extract 2D hair strands from the non-braid region and recover the braid structure. And the final braided hair models are generated by projecting the 2D hair strands and braids on top of the 3D hair volume.

We address tasks related to the image-based 3D hair modeling from 2D hair analysis

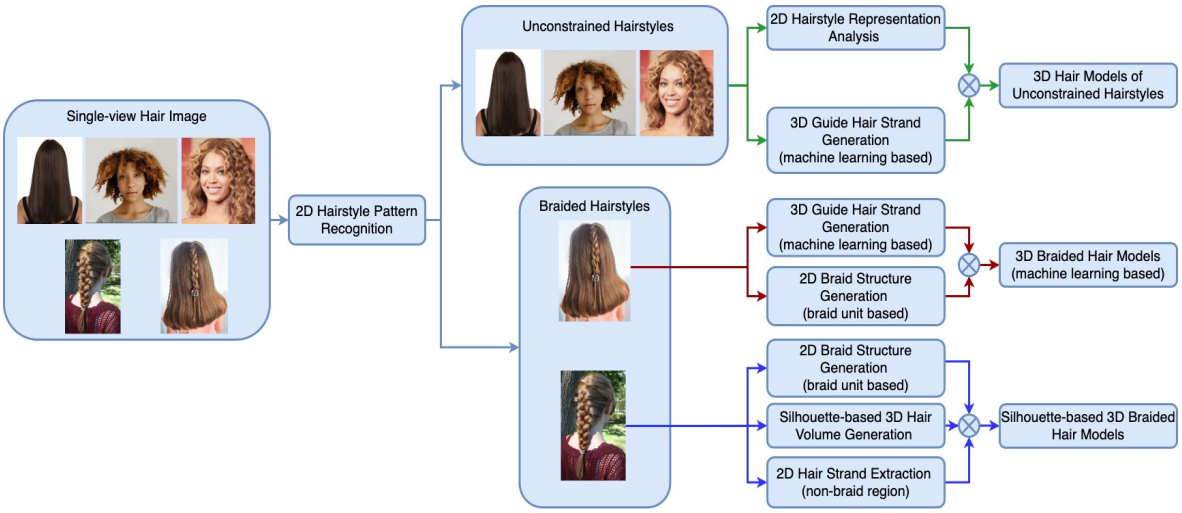


Figure 1.1: Overview of our thesis work.

to 3D hairs reconstruction, including (1) 2D hair strands generation based on hair orientation maps, (2) 2D hairstyle pattern recognition and hair partitioning, (3) hairstyle pattern characteristic learning (for straight, curly, kinky, and braid hairstyles), (4) braid structure discovery based on brain unit identification, (5) deep-learning based 3D hair strand generation, (6) parametric representation of hair shapes, especially for 3D kinky hairs and braided hairs. We approach these tasks by using traditional image processing/computer vision techniques and advanced deep learning models. We also exploit existing datasets and create our datasets to facilitate our 3D hair reconstruction system.

1.5 Contributions

Here are the contributions of this thesis:

- We have developed a system that can generate 3D hair models of a variety of hairstyles from a single-view hair image. And there are no constraints on the views of the input image of our system, unlike existing systems that often require

front face-shoulder images. We also want to emphasize that our reconstructed hair models are not just the 3D volumes of hair, but visually plausible strand-based 3D hair models of a variety of hairstyles.

- Our 2D hairstyle pattern recognition system is the first system that can recognize detailed hairstyle patterns (e.g. straight hairs, curly hairs, kinky hairs, and braided hairs) within the hair region and build the relationship between the recognition results and the parametric representation of 2D/3D hair strands.

- Our system is the first work to tackle the problem of reconstructing two of the most challenging hairstyles given a single-view image. Our work can generate braided hairstyles based on our braid identification method. Furthermore, our work can create visually plausible kinky hairs that existing systems fail to do.

- We have built two hair datasets that can be a good resource for future hair research.

1.6 Organization

The rest of this document is organized into the following chapters:

Chapter 2 provides a comprehensive overview of related systems, including traditional image-based 3D reconstruction methods, deep learning based 3D reconstruction methods, single-view/multi-view image-based hair modeling, hair segmentation system, and existing hair datasets. It provides previous in-depth techniques of general 3D reconstruction and detailed 3D hair modeling.

Chapter 3 elaborates on our contribution regarding 2D hair analysis and discusses the proposed hair strands extraction system, deep learning models for hairstyle

pattern recognition and braid unit recognition, and utilized datasets.

Chapter 4 describes our proposed single-view 3D hair modeling systems, including deep-learning based 3D hair strand generation, 3D braid generation, and 3D kinky hair generation.

Chapter 5 concludes the thesis. It also outlines the directions for future work.

1.7 Published Papers

- Chao Sun, Won-Sook Lee. "Hairstyle based Single-view Hair Structure Generation". ICIP2022. (Submitted)

- Chao Sun, Srinivasan Ramachandran, Eric Paquette, Won-Sook Lee. "Single-view Procedural Braided Hair Modeling through Braid Unit Identification". Computer Animation Virtual Worlds. 2021.

- Chao Sun and Won-Sook Lee. "Braid Hairstyle Recognition based on CNNs". In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications . 2017. pp. 548-555.

- Chao Sun and Won-Sook Lee, "Hairstyle pattern recognition based on CNNs," 2017 IEEE International Conference on Systems, Man, and Cybernetics, 2017, pp. 1840-1845.

- Chao Sun, Zhongrui Li, and Won-Sook Lee. "Reconstruction of Relatively Straight Medium to Long Hair Models using Kinect Sensors". In Proceedings of the 10th

International Conference on Computer Graphics Theory and Applications. 2015. pp. 158–164.

- Zongrui Li, Chao Sun, Won-Sook Lee and I. Kim, "Hair modeling using kinect sensors and DSLR cameras," 11th International Conference on Ubiquitous Robots and Ambient Intelligence, 2014, pp. 22-27.

- Chao Sun, Fatemeh Cheraghchi and Won-Sook Lee, "2D hair strands generation based on template matching". 4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications, 2014, pp. 261-266.

Chapter 2

Literature Review

In this section, we first introduce related works for general objects. We conclude the evaluation metrics and representative datasets. Then we focus on describing related works of 3D hair modeling both from single-view and multi-view. And we also show the representative hair-related datasets.

2.1 Traditional Image-based 3D Reconstruction

Recovering the 3D structure and geometry of objects from one or multiple images has been a fundamental problem in computer vision, and has important applications in 3D modeling, scene recognition and understanding, and robotics. Traditional image-based 3D reconstruction methods are usually stereo methods and Shape-from-X, where X can be motion, shading, silhouette, texture, defocus, etc. **Shape-from-motion** [77], the technique has originally relied on stereo vision, using images pairs of the same scene to reconstruct 3D shapes. It leverages camera motion to estimate camera poses through different temporal intervals and, in turn, estimate depth via triangulation from pairs of consecutive views. **Shape-from-shading** [100], depth can also be acquired by different shooting conditions, such as variations in illumination, however, shape from shading is severely under-constrained. **Shape-from-silhouette** [49] . The silhouette is a set of close contours of an object in an image. It contains some 3D shape information of the object. This method takes the silhouettes that are segmented from each image and corresponding to the camera poses to produce a rough shape of the object, named the visual hull. Traditional multi-view image-based 3D reconstruction methods usually require multiple capture devices and complex capture setups, thus preventing their popularity.

Early machine learning-based 3D reconstruction systems have mainly relied on hand-crafted features and used probabilistic graphical models, and usually made strong assumptions about the object or scene geometry. Recently, with the development of machine learning, especially deep learning, more and more image-based 3D reconstruction systems apply those techniques, such as Convolutional Neural Networks [20], Recurrent Neural Networks [15] , Deep Belief Networks [90], Generative Adversarial Networks [27]

[18] and Deep Convolutional Auto-encoders [26]. Those methods directly learn from data and do not need any prior structural knowledge. Furthermore, large datasets, such as ShapeNet [90], ScanNet [16], Pix3D [75], provide sufficient training and testing data for those deep-learning based 3D reconstruction methods.

2.2 Image-based 3D Reconstruction using Deep Learning

The problem of image-based 3D reconstruction using deep learning can be represented using the following Equation 2.1:

$$\begin{aligned} \operatorname{argmin} \mathcal{L}(I) &= \operatorname{argmin} d(\hat{S}, S) \\ &= \operatorname{argmin} d(f_\theta(I_n), S) \end{aligned} \tag{2.1}$$

- I_n is input. It can be a single-view image ($n = 1$), multi-view images ($n > 1$), and videos.

- S is the target shape, $\hat{S}$ is the reconstructed shape from a predictor f_θ based on input image(s) I_n. Those 3D shapes can be represented in the form of voxel grids, point clouds, etc.

- The predictor f_θ generates the 3D object given the input I_n. Usually, f_θ is composed of an encoder and a decoder. The encoder maps input I_n into a latent space as a feature vector z. The decoder decodes the feature vector z into the desired output in 3D.

- $\mathcal{L}$ is the loss function, it is used to minimize the distance between generated 3D shapes and the target 3D shapes.

In the following section, we will discuss related works in the area of image-based 3D reconstruction using deep learning. Since the formats of the outputs usually have a great impact on the deep learning network architecture, we divide the related works into two categories: volumetric-shape based 3D reconstruction and point-cloud based 3D reconstruction. We also provide a brief summary of some image-based 3D reconstruction methods, as shown in Table 2.1.

2.2.1 Image-based Volumetric Shape Reconstruction

A 3D object can be represented as a probabilistic distribution of binary variables on a 3D voxel grid.1 indicates the voxel is inside the object surface, and 0 indicates the voxel is outside the object [90].

Wu et al. [90] introduce the 3D ShapeNet to represent a geometric 3D shape as a probability distribution of binary variables on a 3D voxel grid, using a Convolutional Deep Belief Network. The model can reconstruct objects from a single-view 2.5D depth map (RGBD camera).

Choy et al. [15] present 3D Recurrent Reconstruction Neural Network (3D-R2N2) which takes in one image or a sequence of images from different viewpoints as input and outputs a voxelized 3D object. Their reconstruction method can be applied with minimal supervision, which means there is no segmentation, key points, or camera calibration needed. Furthermore, the method can successfully perform reconstruction under the conditions of a wide baseline and lacking textures.

Tulsiani et al. [82] present a novel view consistency loss based on a differentiable ray consistency (DRC) term. The DRC term enables the single-view 3D reconstruction system to leverage different types of multi-view observations, such as RGB images, foreground masks, depth, etc. Their system used a decoder composed of 3D up-convolution layers to predict the voxel occupancy probabilities.

Wu et al. [89] propose MarrNet, an end-to-end model that reconstructs 3D shapes from 2.5D sketches. The advantage of using 2.5D sketches as input is that 2.5D sketches are easier to be recovered from 2D images and transferred from synthetic data to real data.

Zou et al. [103] present 3D Primitive Recurrent Neural Networks (3D-PRNN) to perform single depth shape completion. A depth map encoder takes a depth image as input and encodes it into a feature vector. Then a recurrent generator that is made of Long Short-Term Memory units followed by a mixture density network (MDN) [7] takes the feature vector and predicts a set of primitives of the target object.

Smith and Meger [72] present the 3D improved Wasserstein Generative Adversarial Network (3D-IWGAN) which is more stable for training for wider and complex distribution. Then they extend this network to 3D-VAE-IWGAN by integrating VAE-GAN[48] system in the training stage for image-based 3D reconstruction.

Reigler et al. [66] propose the OctNet, a 3D convolutional network that partitions the 3D space into a set of the unbalanced octree. They restrict the maximal depth of an octree to a small number, and this enables 3D convolutional networks that are both deep and high resolution. However, there is a strong assumption that the structure of the individual octree is known. Thus even if the network can reconstruct 3D volumes at a high resolution, it's not flexible since different training is required for different objects.

Han et al. [30] use multi-stage approaches to improve the resolution of generated volumetric. They introduce a local 3D CNN to perform patch-level surface refinement. However, this method requires 3D boundary detection.

Sun et al. [75] present a dataset Pix3D for 3D reconstruction, object pose estimation, and shape retrieval. They also design a new model to perform shape reconstruction and pose estimation. It takes an RGB image as input and predicts its 2.5D sketches, and encodes the 2.5D sketches into a low-dimensional latent vector. Then it decodes the latent vector into a 3D shape and camera parameters.

Tulsiani et al. [81] present a single image-based shape and pose prediction system. Rather than applying direct supervision, the system uses multi-view observations from unknown poses as supervision.

Kundu et al. [47] present an inverse-graphics approach for 3D scene understanding from images. The method can predict the 3D shape and pose of each instance in an image. Given the instance-level 3D representation of the scene, the segmentation and depth map can be easily obtained. In addition, the network can be trained with 2D supervision.

Cao et al. [10] propose the 3D cascaded fully convolutional network (3D-CFCN) to perform high-fidelity volumetric 3D shape reconstruction. The network uses octree 3D deep learning data structure and a multi-stage scheme for detailed shape reconstruction. They propose to use multi-stage network cascades for detailed shape information reconstruction. Using the cascaded structure can reduce the memory request as well as simplify the learning task.

Tatarchenko et al. [79] analyze the recent single-image 3D reconstruction works and conclude that the state of art methods perform recognition rather than reconstruction.

They provide several recommendations for improving the performance of 3D reconstruction.

Yang et al. [96] propose 3D-RecGAN++, which reconstructs the complete 3D structure of a given object from a single arbitrary depth view using generative adversarial networks. The network only takes the voxel grid representation of a depth view of the object as input and can generate the complete 3D occupancy grid with a high resolution of 2,563 by recovering the occluded/missing regions. The key idea is to combine the generative capabilities of the 3D encoder-decoder and the conditional adversarial networks framework, to infer accurate and fine-grained 3D structures of objects in high-dimensional voxel space.

Xie et al. [93] propose Pix2Vox, a single-view and multi-view 3D reconstruction framework. The network uses an encoder-decoder structure. It first generates a coarse 3D volume from each input image. Then the high-quality reconstructions of each part from different coarse 3D volumes are selected and fused by a context-aware fusion model. Then the fused 3D volume is further refined to generate the final output.

In conclusion, most of the systems apply the encoder-decoder network architectures to generate 3D shapes . The networks encode the input image into a latent vector z and then decode z into the voxel representation of the 3D shape. This can be done under the supervision of volumetric 3D data or 2D image only. However, the main disadvantage of volumetric representation for 3D shapes is that it can not capture the smooth surfaces or the geometric structures inside the object.

Table 2.1: Summary of some representative image-based 3D reconstruction methods. GT: ground truth, CD: chamfer distance.

Method	**Input**		**Network**	**Supervision**	**Output**		**Performance**	
	Train	Test			Type	Resolution	IoU	CD
Choy et al. [15]	Multiple images, 3D GT	Single image	encoder-decoder, LSTM	3D	Volumetric	32^3	$(0.56, -, 0.537)@32^3$	(-,-,-)
Tulsiani et al.[82]	Multiple images, Silhouette, Depth	Single image	encoder-decoder	2D	Volumetric	32^3	$(0.546, -, 0.536)@32^3$	(-,-,-)
Wu et al.[89]	Single image, 3D GT	Single image	encoder-decoder, TL, 3D-VAE	2D, 2.5D	Volumetric	32^3	$(0.57, -, 0.39)@128^3$	(-,-,-)
Fan et al.[22]	Single image, 3D GT	Single image	encoder-decoder	3D	Point Cloud	1024	$(0.64, -, -)@3263$	$(5.62, -, -)$
Tulsiani et al.[81]	Multiple images, Segmentation	Single image	Encoder, Decoder, PoseCNN	2D	Volumetric + pose	-	$(0.62, -, -)@32^3$	(-,-,-)
Mandikal et al.[59]	Single image, 3D GT	Single image	3D-VAE TL-embedding	3D	Point Cloud	2, 048	(-,-,-)	$(5.4, -, -)$
Jiang et al.[42]	Single image, 3D GT	Single image	encoder-decoder, GANs	3D	Point Cloud	1, 024	$(0.7116, -, -)@32^3$	(-,-,-)
Xie et al. [93]	Single/Multiple image(s), 3D GT	Single/20 image(s)	Encoder,Decoder, Refiner	3D	Volumetric	32^3	$(0.658, -, 0.669)@32^3$	(-,-,-)
Mandikal and Babu [58]	Single image, 3D GT	Single image	Conv+FC layers	3D	Point Cloud	16, 384	(-,-,-)	(8.63,-,-)

2.2.2 Image-based Point Clouds Reconstruction

Compared to volumetric data, a 3D point cloud is still not a typical object in the deep learning community. However, compare to volumetric 3D grids, point clouds can represent the object surfaces more conveniently and efficient [22]. And point clouds are easy to capture using depth-sensing devices, thus it is possible to create ground-truth datasets. In addition, it is easier to perform object rotation, translation, and scaling since we know the coordinates of the point clouds.

Tatarchenko et al. [78] present a convolutional network capable of inferring a 3D representation of a previously unseen object given a single image of this object. The network can predict an RGB image and a depth map of the object as seen from an arbitrary view. Then several of these depth maps are fused to give a full point cloud of the object. The system successfully deals with objects on the cluttered background and generates reasonable predictions for real images.

Qi et al. [64] propose PointNet, which directly processes point sets and provides a unified architecture for object classification, part segmentation, and scene semantic parsing. The key idea of PointNet is to learn a spatial encoding of each point and then aggregate all individual point features to a global point cloud signature. PointNet does not capture the local structure induced by the metric. However, exploiting local structures has proven to be important for the success of convolutional architectures.

Fan et al. [22] build a conditional generative network, PointOutNet, to reconstruct 3D point clouds from a single image. The conditional shape sampler is capable of predicting multiple plausible 3D point clouds from an input image.

Mandikal et al. [59] propose 3D-LMNet, a latent embedding matching approach for

3D reconstruction. The method contains two stages: (1) a 3D point cloud auto-encoder was used to learn the 3D point cloud latent space and then (2) an image encoder was used to map the 2D image to the learned latent space.

Jiang et al. [42] present the geometric adversarial loss (GAL) as a global constrain for single-view image based 3D point cloud reconstruction. The GAL contains two components: geometric loss and conditional adversarial loss. The former ensures the constructed model close to the ground truth from different views. The latter generates a sentimentally meaningful point cloud.

Mandikal et al. [58] introduce DensePCR, a deep pyramidal network for point cloud reconstruction that hierarchically predicts point clouds of increasing resolution. The proposed network first predicts a low-resolution point cloud, and then hierarchically increases the resolution by aggregating local and global point features to deform a grid.

Wang et al. [83] propose the MVPNet, a generative network that reconstructs an object's surface from a single image. The object's 3D surface is defined as an aggregation of dense partial point clouds from different views. Each point cloud is parametrized in a regular 2D grid aligned on an image plane of a viewpoint. Then an encoder-decoder network is used to generate the multi-view dependent point clouds from a single image by regressing their 3D coordinates and visibilities.

To sum up, point-based representations can deal with 3D shapes of arbitrary topologies. However, those methods often requires post-processing steps to retrieve the 3D surface mesh. Thus, it is difficult to train an end-to-end system to generate the final mesh directly from the input.

2.2.3 Evaluation Metrics

The following methods are the most commonly used metrics to calculate the accuracy of the 3D reconstruction algorithms.

- **Intersection over Union (IoU)** The IoU measures the ratio of the intersection between the volume of the predicted shape and the volume of the ground-truth shape.

$$IoU = \frac{S_{recons} \cap S_{gt}}{S_{recons} \cup S_{gt}} \tag{2.2}$$

  where S_{recons} is the reconstructed volume and S_{gt} is the ground truth volume. The higher the IoU value, the better the reconstruction of a 3D model.

- **Cross Entropy Loss** The cross entropy loss is defined as:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\left\{p_i \log \hat{p}_i + (1-p_i)\log(1-\hat{p}_i)\right\} \tag{2.3}$$

  where p_i is the ground-truth probability of voxel i occupied, $\hat{p}_i$ is the estimated probability, and N is the number of voxels.

- **The Mean Squared Error** Let S_{gt} be the ground truth 3D shape and S_{recons} the reconstructed shape. The mean squared error is defined as the symmetric surface distance d_{3d} between the reconstructed shape S_{recons} and the ground truth shape. d_{3d} is computed by sampling points on the surfaces and using normalized points distance to estimate the similarity.

$$d_{3d} = \frac{1}{N_{gt}}\sum_{p \in S_{gt}} d(p, S_{recons}) + \frac{1}{N_{recons}}\sum_{\hat{p} \in S_{recons}} d(\hat{p}, S_{gt}) \tag{2.4}$$

  where N_{gt} is the number of sampled points on S_{gt} and N_{recons} is the number of sampled points on S_{recons}. Smaller MSE means better reconstruction.

- **Chamfer Distance (CD)** For each point in each cloud (S_{gt} and S_{recons}), CD finds the nearest point in the other point set, and sums up the distances.

$$CD(S_{recons}, S_{gt}) = \sum_{p \in S_{gt}} \min_{q \in S_{recons}} \|p - q\|_2^2 + \sum_{q \in S_{recons}} \min_{p \in S_{gt}} \|p - q\|_2^2 \tag{2.5}$$

- **Earth Mover's Distance (EMD)** Consider $S_{gt}, S_{recons} \subseteq R^3$ of equal size $s = |S_{gt}| = |S_{recons}|$. The EMD is defined as:

$$EMD(S_{recons}, S_{gt}) = \min_{\phi: S_{recons} \to S_{gt}} \sum_{p \in S_{recons}} \left\|p - \phi(p)\right\|_2 \tag{2.6}$$

where $\phi : S_{recons} \to S_{gt}$ is a bijection. However, in practice, EMD is very expensive to calculate, thus researchers[22] implement a $(1 + \epsilon)$ approximation scheme given by [6].

It is suggested that metrics like Chamfer distance, and Earth Mover's distance can reflect human perception better than IoU, although many 3D reconstruction papers use IoU as their similarity measurement between the reconstructed shape and the target shape [75].

2.2.4 Datasets

Unlike traditional 3D reconstruction methods, deep-learning based 3D reconstruction methods require a large amount of annotated data for training and evaluation. The data should be in the form of images and their corresponding 3D data, which is very challenging to obtain. A summary of datasets that have been used in the literature is shown in Table 2.2. The datasets for image-based reconstruction methods usually contain one/multi images of real or synthesized scenes, captured with calibrated cameras, and their corresponding 3D information as ground truth.

Table 2.2: Commonly used datasets for deep-learning based 3D reconstruction

Dataset	Image		3D data			Camera parameters
	No	Type	No	Type	Environment	
NYU V2[71]	1,449	Real	35,064	3D point cloud, Depth	indoor	Intrinsic
KITTI2[25]	41,788	Real	40,000	3D point cloud	outdoor	Intrinsic, extrinsic
IKEA[43]	759	Real	219	3D model	indoor	Intrinsic
PASCAL 3d+[92]	30,899	Real	360,004	3D model	indoor, outdoor	Intrinsic, extrinsic
ShapeNet[90]	-	Rendered	127,915	3D model	-	Intrinsic
Stanford 2D-3D-S[4]	70,496	Real	6,005	3D point cloud	indoor	Intrinsic, extrinsic
ObjectNet3D[91]	90,127	Real	44,174	3D model	indoor, outdoor	Intrinsic, extrinsic
ScanNet[16]	2,492,518	Real	36,123	Depth	indoor	Intrinsic, extrinsic
Pix3D[75]	9,531	Real	1,015	3D model	indoor	Focal length, extrinsic

2.3 Image-based 3D Hair Modeling

Hair is one of the most challenging objects to capture using standard computer vision techniques. Researchers tried to reconstruct the 3D hair models from captured real-life hair images. Image-based 3D hair modeling can be divided into different categories as follows:

- **Multi-view vs. Single-view** Multi-view image based 3D hair model reconstruction has been widely researched. However, this kind of system always requires large amounts of input images to reconstruct 3D hair volume and 2D/3D orientation fields. The capture systems always require multiple cameras, light sources, and a complex setup. On the contrary, single-view hair reconstruction usually requires a simple capture device (such as a hand-held camera). However, traditional single-view hair modeling methods can only reconstruct the model of the reference view, the hidden view can not be reconstructed. Recently, combined with 3D hair model data and deep learning techniques, single-view hair modeling can reconstruct complete 3D hair models that are visually plausible as well as physically plausible.

- **Orientation-field based vs. Data-driven based** The orientation-field based hair modeling methods focus on generating realistic 3D orientation fields, then tracing the 3D hair strands based on the 3D orientation fields. Since the input usually contains a large number of images, the reconstruction results are more realistic. The data-driven based hair modeling methods rely on the following main components: 3D hair datasets, example strands generation, 3D hairstyles retrieval, and hairstyles combination, hair model refinement. Thus the size and diversity of the 3D hair datasets, the storage space, the efficiency of the retrieve, and refinement algorithms have a great impact on the performance of data-driven methods.

- **Static vs. Dynamic** Most existing hair modeling systems use static images as input. The static hair modeling can reconstruct various hairstyles, and the reconstructed results are more realistic. On the contrary, dynamic hair modeling is very challenging due to the complexity of hair motion and geometry, changing occlusion among hair strands, motion blur, etc. Thus the hairstyles that can be handled by dynamic hair modeling are very limited (mostly straight hairstyles).

2.3.1 Multi-view Image based Hair Modeling

Paris et al. [62] present a method to capture the real geometry of a person's hair from multi-view images. The capture system consists of a fixed camera and a light source moving along a predefined trajectory around a person's head. The 2D orientation map from one viewpoint is calculated using orientation filters based on input image sequences. Then the 3D orientation maps are generated by analyzing the light reflected by the hair fibers. The 3D orientation maps guide the growth of hair strands. In order to perform

a full hair model reconstruction, this procedure is repeated from different viewpoints and the corresponding results are merged. However, there are some limitations of this method: (1) the capture system is very complex, as shown in Figure 2.1-(a). (2) It needs intensive user interactions. (3) The hidden hair strands can not be captured. (4) Thick strands (such as dreadlocks) or short hair pointing toward the camera can not be reconstructed. The capture system and reconstructed hair model are shown in Figure 2.1.

(a) **(b)** **(c)**

Figure 2.1: Hair capture system and reconstructed hair model [62].(a) The Capture system. (b) One reference image. (c) The reconstructed hair model shown in the same view as the reference image.

Wei et al. [88] used a handheld video camera to capture 2D hair images under natural lighting conditions. They detected the local orientation of every pixel in each hair image and represented every hair fiber using a sequence of chained line segments. Then they applied triangulation for each fiber segment using image orientations of multiple views to reconstruct the 3D hair fibers. They generated a visual hull to constrain the synthesis of hair fibers. This method does not require a special capturing system with controlled illuminations compared to [62]. However, it still needs user interactions, such as hair region segmentation. The reconstruction results are shown in Figure 2.2.

Paris et al. [63] present an active acquisition system called Hair Photobooth. The system is composed of 16 video cameras, 150 LED light sources and 3 projectors. The hair images are captured under different lighting directions from a number of cameras and are used to recover the hair reflectance field. Triangulation is used to retrieve the

(a) (b) (c)

Figure 2.2: Multi-view hair modeling results of [88].(a) The image captured by the video camera. (b) The rendering result with recovered diffuse colors. (c) The rendering result with a constant hair color.

location of hair strands. This method can capture accurate data on the exterior hair strands. The hair model is generated from the scalp to the captured exterior hair layer under the constraints of orientation fields. The capture system and hair reconstruction results are shown in Figure 2.3.

(a) (b) (c) (d)

Figure 2.3: Capture system and reconstruction results of Hair Photobooth [63]. (a) The capture system. (b) The reference photo (not input data). (c) The reconstructed geometry. (d) The rendering result.

Luo et al. [55] propose a multi-view hair reconstruction algorithm based on orientation fields with structure-aware aggregation. The system only requires consumer-level hardware and produces feasible results. The hair orientation (which can be computed using a bank of filters) is a stable feature across nearby viewpoints and can be used as a reliable matching criterion. The method of computing 2D hair orientation is used in multiple hair modeling papers. However, the proposed method doesn't reconstruct

stand-based hair models. The capture system and the reconstruction results are shown in Figure 2.4.

(a) **(b)**

Figure 2.4: Multi-view hair reconstruction results of [55]. (a) The hair capture system. (b) The input images and corresponding reconstruction results.

Luo et al. [56] present a structured-aware hair model reconstruction system. Local strand segments are extracted from the point cloud (calculate by patch-based multi-view stereo [24]) and orientation field (calculated based on multiple still images). Then those hair strand segments are clustered into 'ribbons' and connected across gaps in the point cloud. At last, those ribbons are connected as well as the scalp based on a global direction analysis to get complete wisps. The capture system and reconstruction results are shown in Figure 2.5.

Hu et al. [37] propose a data-driven method to automatically reconstruct braided hairstyles. A 3D braid patch dataset is generated using procedure modeling. The common braided hairstyle is represented using sinusoidal functions. The generated braid patches are shown in Figure 2.6-(a). The hair reconstruction procedure is: the local 3D orientation field is calculated as in [56] based on the input 3D point clouds. Then the input braid point cloud is aligned with the patches in the 3D braid patch datasets based on random sampling fitting. Users are required to choose a similar braid model from

(a) (b)

Figure 2.5: The capture system and reconstruction results of structured-aware hair modeling method [56]. (a) The capture system for wigs and real people, (b) the reference images and rendering results.

the dataset or to provide a rough initial scale to help speed up the searching. Then the optimal braid patches are calculated and the center lines of the braid patches are connected to generate complete braids. At last, the non-braid region and the braid region are combined to generate the complete hair model. The reconstructed braid models are shown in Figure 2.6.

(a) (b)

Figure 2.6: Braid hairstyle capture [37]. (a) The braid patch samples. (b) The reconstructed full hair models.

Hu et al. [35] introduce a hair capture system using simulated examples. 18 hair model datasets are generated by applying the super-helices model to simulate static hair strands. The generated strands are then fitted with the cover strands that are reconstructed from

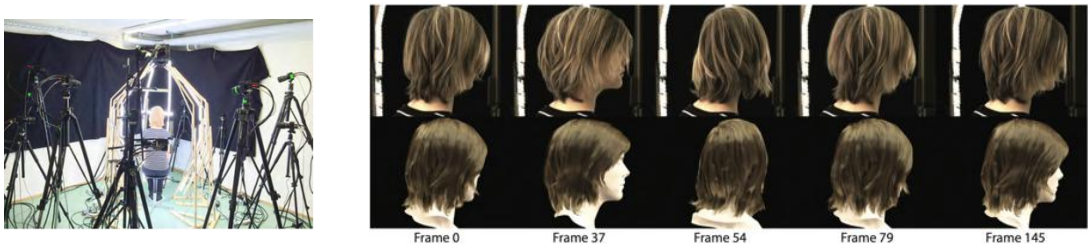
multi-view stereo hair images to obtain structure plausible hair models. By introducing the simulated examples, the procedure of manual cleaning up for the outliers can be avoided. However, the strand fitting is very time-consuming since it needs to go through all datasets to retrieve the closest hair example and combine them to match the input hairstyle. In addition, user strokes are needed to deal with some simple constrained hairstyles, such as the ponytail. The reconstruction results are shown in Figure 2.7.

(a) **(b)**

Figure 2.7: Hair reconstruction results of robust hair capture using simulated examples [35]. (a) The reconstruction results of unconstrained hairstyles. (b) The reconstruction result of a ponytail hairstyle. The green lines are user strokes.

Hu et al. [34] present a method to produce both the geometry of a hairstyle and estimated simulation parameters. This method is a data-driven parameter estimation method rather than an algorithm that focuses on capturing accurate dynamic hair. The hairstyle can be easily edited by changing the head motion, re-simulating in completely different environments, or adding additional external forces. The capture system and experiment results are shown in Figure 2.8.

Liang et al. [52] proposed a method that takes a video of a person's head in the wild as input and outputs a detailed 3D hair strand model combined with a reconstructed 3D head to produce a full head model. The video frames are processed to create a structure-from-motion model, estimate camera poses, per-frame depth, and compute a



(a) **(b)**

Figure 2.8: Simulation-ready hair capture system [34]. (a) The image capture system. (b) The captured dynamic hair.

rough visual hull of the person with view confidence. Then a hair segmentation classifier is trained to segment the hair region in each frame and helped to extract the hair part out of the visual hull. Furthermore, the hair directional classifiers are trained to label and predict the hair direction in the hair pixels of each video frame. Then the 2D hair orientation estimation method is applied to each video frame to extract 2D hair strands. Each video frame has an estimation of 2D strands, those strands are projected to the per-frame depth to estimate 3D strands, and around 100 3D strands are obtained after a merging procedure. Then the recovered 3D strands are used to query a database of hair models and adjust the retrieved matches with global and local deformations to create a full hair model. However, this method can not work on highly dynamic hairstyles, curly hairstyles, or complicated hairstyles such as braids. Furthermore, the method can not be used on videos with complicated backgrounds (multiple people in the backgrounds). Moreover, this method requires an input video covering a view range of at least 90°. The experimental results are shown in Figure 2.9.

Yang et al. [97] introduce a dynamic hair modeling framework based on deep learning. The framework is composed of two neural networks: HairSpatNet and HairTempNet. The HairSpatNet is used to infer 3D occupancy fields (for hair volume) and 3D orientation

Figure 2.9: 3D hair reconstruction results from video [52].

fields (for hair growing directions) of hair geometry from 2D images, and HairTempNet is used for extracting bidirectional 3D warping fields of hair motions from video frames. The spatial and temporal features predicted by the networks are subsequently used for growing hair strands with both spatial and temporal consistency. Experiments demonstrate that the method is capable of constructing plausible dynamic hair models that closely resemble the input video. The experimental results are shown in Figure 2.10.

(a) **(b)**

Figure 2.10: Dynamic hair model reconstruction results from [97]. (a) The input frames. (b) The corresponding reconstructed hair models.

2.3.2 Single-view Image based Hair Modeling

Chai et al. [14] present a system to create a plausible high-resolution strand-level 3D hair model based on single-view image. First, a 3D morphable head model [8] is fitted to the person in a portrait image, and the hair region is annotated by the user using a few strokes. Then a spare set of 2D hair strands are traced within the hair region.

By applying the optimization of the depth constrain, local layering constraints, and regularization terms, the sparse 3D hair strands can be obtained. The sparse 3D hair strands are used to define a bounding volume around the head and more 3D hair strands are generated to fill the occluded region in the hair images. In addition, there are some applications based on the generated 3D hair model, such as portrait pop-ups, hairstyle replacement, and hairstyle editing. However, the method suffers from some limitations: (1) the generated hair model is not a full 3D model of the entire hair volume, (2) the system fails when there are not enough distinguishing features, (3) the generated hair strand segments are not connected to the scalp, thus the hair model can not be animated. The reconstruction results are shown in Figure 2.11.

(a) **(b)** **(c)** **(d)**

Figure 2.11: 3D hair model reconstruction results from single portrait image [14]. (a) The input image with user strokes. (b) The portrait pop-up rendered in the original view. (c), (d) The portrait rendered in two novel views.

Chai et al. [13] improve the method [14] to generate a 3D hair model that is both visually and physically plausible (the hair roots are fixed on the scalp). Unlike [14], the non-directional 2D hair segments will be used as constraints instead of the real 2D hair strands, and the arc sampling method [5] is applied to decide the direction of 2D hair strands. They also propose a two-step approach to generate the final individual 3D

strands. First, sparse initial hair strands are generated from the scalp surface following the 3D direction field. Second, additional strands are synthesized and refined over all hair models. In addition, the hair modeling method has been extended to create dynamic hair models from video clips with modest hair motion and change the hair's geometric properties. However, their reconstructed hair models are only visually plausible under the original viewpoint. The reconstruction results are shown in Figure 2.12.

(a) **(b)**

Figure 2.12: 3D hair model of [14]. (a) The input image. (b) The rendered 3D reconstructed hair model.

Hu et al. [36] present a system that creates a high-quality 3D hair model from a single input reference photo using a database of full 3D hairstyles and user strokes. The method contains a coarse-to-fine modeling strategy and adopts a hierarchical two-level representation to accurately model the target hairstyle. At the coarse scale, the example hairstyles that best match the 2D user strokes are retrieved from the database. The retrieved examples are combined into a 3D orientation field based on the user strokes and local orientation consistency in a Markov Random Field framework. The combined 3D orientation field is used to produce the structure of the target hairstyle by growing strands from the scalp of the fitted 3D head model. Then the combined hair strands are deformed to match the 2D orientations of the target hairstyle to reach a more detailed

look, as shown in Figure 2.13. This method is the first data-driven method that can generate a full 3D hairstyle from a single image. However, the quality of the generated3D hair models is highly dependent on the 3D hair dataset, and the quality and accuracy of user strokes.

(a) (b) (c) (d) (e)

Figure 2.13: Reconstructed hair model [36]. (a) The reference image. (b) The user strokes. (c) The hairstyle combination result. (d) The combined hairstyle. (e) The final 3D hair strands.

Chai et al. [11] present a system that can model high-quality hairstyles. The system combined the base shape, the normals estimated by shape from shading, and the 3D helical hair prior to reconstructing high-quality 3D hair models. The final depth map was calculated by minimizing an energy function, which contains the energies of the base shape, shape from shading normals, and the helical hair prior. The reconstructed 3D hair models can be used for high-quality portrait relighting and 3D-printed portrait reliefs. The reconstruction results are shown in Figure 2.14.

Chai et al. [12] present an automatic hair modeling method from a single portrait image. The input portrait image is classified into a few precomputed hair spatial distribution classes and then generates a hair segmentation and a hair growth direction map using a neural network. Then a few 3D hair model candidates are retrieved from a large 3D hair exemplar dataset. Those models then are deformed and refined. The final 3D hair strands are generated based on the closest matching candidate and the direction

(a) **(b)** **(c)**

Figure 2.14: High-quality hair modeling [11]. (a) The input image and user strokes. (b) The 3D hair model. (c) The side view of the 3D hair model.

map of the images. The generated hair model can be applied to portrait manipulation, physically-based animation, etc. The reconstruction results are shown in Figure 2.15. However, this method can only be used on portrait images with faces. Furthermore, the capacity of this method depends on the size and diversity of the 3D hair datasets. For instance, constrained hairstyles such as braids can not be created. Figure 2.15 -(b) shows a reconstruction result for a curly hairstyle, however, the global appearance is similar, however, the details need to be improved. Especially, in the second row, the deformed matching hair shape and the final hair model have different structures.

Zhang et al. [98] introduce a four-view image-based hair modeling method. First, the hair regions are segmented from four hair images taken from the front, back, left, and right views. Then a 3D hair model is selected from a predefined hair model database that best matches the hair contours at different views. Then the hair model is deformed to align with the hair contour and clipped against the hair-face boundaries. Furthermore, the hair texture is synthesized over the hair shape from the hair images. The 3D hair direction field is computed based on the hair shape surface over the entire volume. At last, the 3D hair strands are generated according to the directional field and refined by piecewise helix fitting. However, this work still needs some manual work for hair region segmentation

(a) **(b)** **(c)**

Figure 2.15: Hair capture results of [12]. (a) The input images. (b) The deformed matching hair shapes. (c) The final hair models.

and camera parameter configurations. It can not handle constrained hairstyles, such as braid or kinky hair. The reconstruction results are shown in Figure 2.16.

(a) **(b)**

Figure 2.16: Reconstructed hair model in [98]. (a) The multi-view input hair images. (b) The reconstructed 3D hair model from different viewpoints.

Hu et al. [38] present an automatic system that can reconstruct a full 3D head model from a single unconstrained image. In order to reconstruct the hair model, semantic hair attributes (length, level of baldness, hairlines, fringes) are extracted using the deep

residual network. These attributes are compared with a large hairstyle dataset. After the comparison, a small hair model dataset of relevant models is obtained. In this reduced hair model dataset, the closet hairstyle to the input image is selected based on the similarity of the hair silhouette and the orientation field. At last, a mesh fitting step is applied to deform the hairstyle model to make the appearance similar to the input image. The results are shown in Figure 2.17. The limitation of this method is that the wrong hairstyle model can be retrieved because of the view-occlusion and insufficient hairstyle samples in the dataset.

(a) **(b)** **(c)**

Figure 2.17: Hair capture results of [38]. (a) The input images. (b) The face mesh and hair polystrips. (c) 3D avatars.

Zhou et al. [102] present a deep learning-based method HairNet to generate full 3D hair geometry from an unconstrained image. The entire pipeline contains three steps: firstly, PSPNet [101] is adopted to produce a pixel-wise hair mask of the input image, followed by computing the undirected 2D orientation field of the hair region using Gabor filters. Secondly, the HairNet encodes the 2D orientation fields to a latent vector, then decodes the target 3D hair strands from the vector-based on a multi-scale mechanism.

Finally, a dense and smooth hair model is generated. The experimental results show that HairNet can generate hairstyles of arbitrary resolution while maintaining a natural appearance. In addition, HairNet is about three times faster than [12] and only uses a small amount of storage space. The experimental results are shown in Figure 2.18. However, HairNet fails to generate kinky, afro, or buzz-cut hairstyles. Also, there are no experimental results showing that HairNet can generate braids.

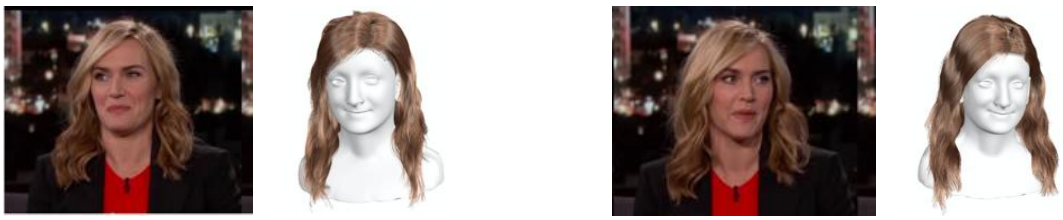


Figure 2.18: 3D hair reconstruction results of [102].

Saito et al. [69] present an end-to-end single view 3D hair synthesis method based on volumetric variational autoencoder [46] and a hair embedding network. The system uses the volumetric occupancy and corresponding flow field to represent 3D hairstyles. The volumetric occupancy indicates the hair model volume, and the flow field shows the growth direction of 3D hair strands. However, post-processing is needed, such as hair mask segmentation and head model digitalization based on input image [38], hairstyle fitting to head model [12] alignment between the hair model to hair segmentation mask [38], matching local details of synthesis stands to the 2D orientation map from input images [36]. However, since there's compression during the processing of the encoder and the information loss in latent coefficient alignment, their results lack corresponding details between the input image and the output hair structure. The 3D hair reconstruction results are shown in Figure 2.19. Figure 2.19-(c) shows the generated 3D hair model without post-processing. The model only shows the general volume of the hair and lacks

details. In order to create a very curly hairstyle such as Figure 2.19-(f), post-processing is needed. Otherwise, the generated hair strands will be relatively straight if only follow the volume and growth direction in Figure 2.19-(e).

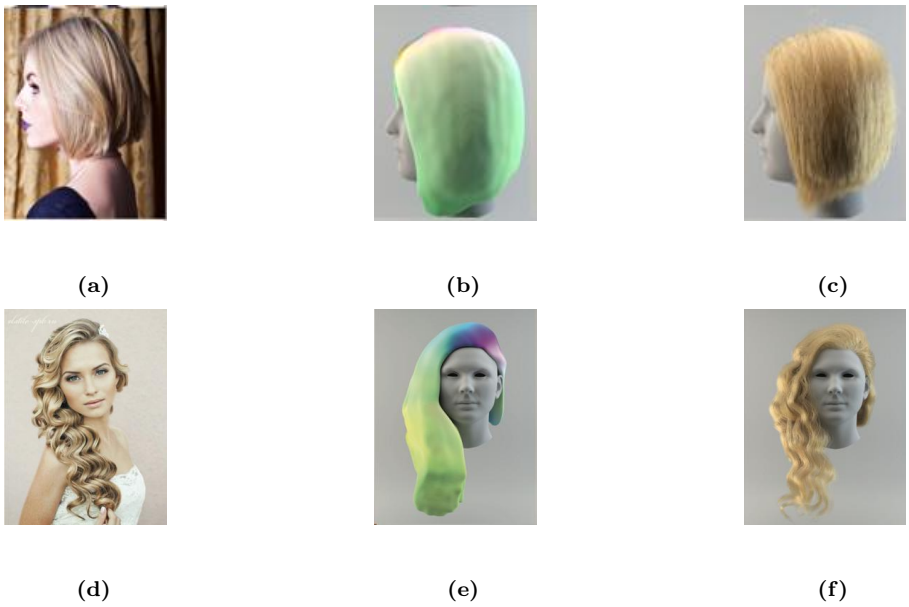


Figure 2.19: 3D hair strand reconstruction results [69]. (a), (d) The input images. (b), (e) The predicted volumetric representation with color-coded local orientations. (c), (f) The generated 3D hair strands.

Zhang and Zheng [99] present HairGANs, a single-view hair modeling method based on generative adversarial network [28]. The input image and a bust model are aligned [9] first. Then the hair orientation maps, the orientation confidence maps, and the bust depth map are computed based on the aligned model. Those are the inputs of the network. The 3D volumetric field is computed by the generator network as the guidance for hair synthesis. The experiment shows that the generated hair model resembles the hair input image and also keeps details in other views.

Xing et al. [94] propose a VR-based interactive hair modeling system. The system takes user gestures as key strips and estimates the user-intended hairstyle based on a

nonlinear classifier trained by a deep neural network. The predicted hairstyle will match the input strips. The interactive hair modeling is shown in Figure 2.20. The green strokes in the upper row of Figure 2.20 are the key stripes drawn by the users, the bottom row shows the corresponding changing in the 3D hair models.

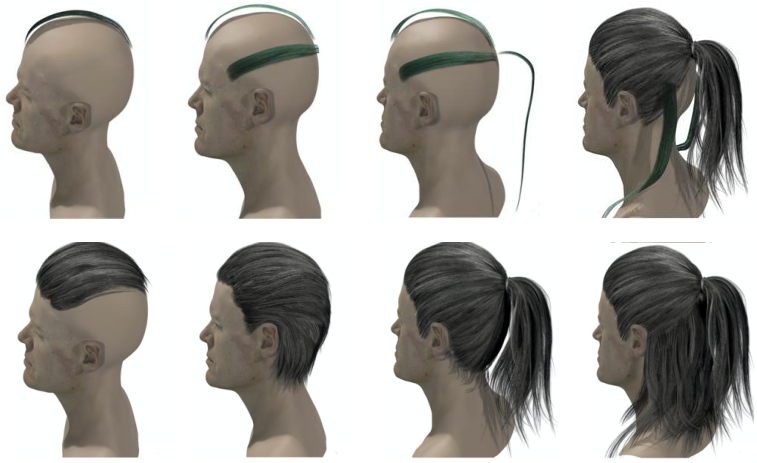


Figure 2.20: VR-based interactive hair modelling system [94].

Shen et al. [70] proposed DeepSketchHair, an interactive hair modeling system from 2D sketches based on deep learning, as shown in Figure 2.21. This method contains three deep learning networks: S2ONet, O2VNet, and V2VNet. The S2ONet is used to predict a dense 2D orientation field based on the user's sketches. The O2VNet is used to combine the 3D bust model and the 2D orientation field and produce a 3D vector field. The 3D vector field is used to guide the growth of 3D hair strands. Based on the generated 3D hair model, the users can also edit the 3D hair model based on V2VNet by drawing new sketches from new views.

(a) (b) (c) (d)

Figure 2.21: DeepSketchHair [70]. (a) The 3D bust model and the user sketches of the hair contour (red) and hair strand directions(blue). (b) The generated hair model. (c) Users modify the hair model with new sketch from new view. (d) The modified hair model.

2.4 Hair Region Segmentation

2D hair region segmentation is an important step in both 2D hair image processing and 3D hair modeling procedure. 2D Hair segmentation methods can be divided into two categories: hand-crafted features based segmentation methods and deep neural network based segmentation methods.

2.4.1 Hand-crafted Feature based Hair Region Segmentation

Early hair region segmentation methods are usually based on hair colors, frequency, and location information.

Yacoob and Davis [95] develop a pixel-wise hair detection method. The location information and color information of the face, eyes, skin, and hair are used to determine the hair pixel in the front face images. The hair segmentation accuracy is 71%. Lee et al. [50] propose an automatic hair and face segmentation method using Markov Random Field (MRF) based on the color and location information of face, hair, and background regions. Aarabi [1] present a heuristic-based method to detect and segment hair by

using the information extracted from face position, hair color, skin color, skin mask, and background color. The reported segmentation accuracy is 75%. Khan et al. [44] reached 93% of the hair classification accuracy by using more features with random decision forests. Wang et al. [85] propose a compositional exemplar-based model (CEM) for hairstyle segmentation. Wang et al. [86] present a part-based model robust to hair shape and environment variations. The key idea is the proposed subspace clustering dependency (SC-Dependency) which can be used to estimate the co-occurrence probabilities between local shapes.

2.4.2 Machine Learning based Hair Region Segmentation

Recent works based on deep learning models achieved good performance. However, most of the methods have constraints on the input, for example, the following methods all require the input to be head-shoulder images. Guo and Aarabi [29] present a method for binary classification using neural networks with the help of a pre-trained heuristic classifier. The heuristic classifier is used to segment high-confidence positive hair patches from images. Those high confidence sets are used to train the neural networks to classify the low-confidence set hair patches. Qin et al. [65] employ fully convolutional networks for hair segmentation. However, a post-processing method based on dense conditional random fields is needed because of the coarseness of fully convolutional network segmentation results. Levinshtein et al. [51] presented a real-time hair matting method. They modified the MobileNet [33] architecture into a fully convolutional network for hair segmentation - HairSegNet. They also developed the HairMatteNet based on convolutional neural networks using skip connection and adding a new loss function. The method reports an on-par performance with best performing results in [65], and performances

better than [29]. Ma et al. [57] develop a deep learning-network based approach that employs both time-of-flight (ToF) depth map and the RGB gradient maps to produce an initial hair segmentation with labeled hair components. Then the result is refined by imposing ToF noise prior under the conditional random field. They also produce an RGBD hair dataset with 20k+ head images captured on 30 subjects with a variety of hairstyles at different view angles. We have tested some images from [57] using the network from [51] and we find that [51] can provide better segmentation results. Ileni et al. [3] used a variant of the U-Net fully convolutional network [67] to detect hair pixels. Then they use the flood-fill based method to fill the gaps within the segmentation mask.

Sun and Lee [74] proposed a convolutional-neural-network based hairstyle pattern recognition system that shows good recognition performance for different hairstyles. The method is also efficient due to the pre-segmentation of the hair region using superpixels. Muhammad et al. [60] proposed a hair image dataset along with a hair segmentation method. Their method can successfully detect hair regions. However, they perform a sliding window recognition operation on the whole image, which is computationally very expensive.

2.5 Datasets

In this section, we summarize the existing datasets for hair modeling or hair-related research. From Table 2.3 we can find that:

- Compare to other 3D reconstruction tasks, there are only limited datasets for 3D hair modeling. Some of the datasets are not designed for 3D hair modeling.
- Most of the 3D hair models are synthetic models that were created manually or

generated using existing 3D hair modeling methods.

- Most of the 3D datasets don't have real 2D images aligned with real 3D hair models.

Note: The authors[69] first collect all 343 3D hair models from USC-HairSalon. Then they generate 816 new hair models. And they flipped all 1,159 3D hair models horizontally to obtain the final 2,318 3D hair models. 3DHW[12] is not open-source.

Table 2.3: Some Datasets that are used for 2D hair analysis and 3D hair reconstruction.

Dataset	**Image**				**3D data**			**Hairstyle**
	No.	**Type**	**Obj No.**	**Description**	**No.**	**Type**	**Representation**	
LFW[39][40]	13,233	Real	5,749	Front view	-	-	-	Multi
USC-HairSalon[**10.1145/2766931**]	-	-	-	-	343	Synthetic	Strand, Polygon wisps	Multi, no kinky/braid
CelebA[54]	202,599	Real	10,177	Front view	-	-	-	Multi
HairstylePattern[74]	4,000	Real	232	Patches	-	-	-	Multi
Figaro-1K[**umar2018hair**]	1,050	Real	-	Multi-view	-	-	-	Multi
Patch-1K[**umar2018hair**]	2,100	Real	-	Patches	-	-	-	Multi
3DHW[12]	20,000	Real	-	Front view Hair distribution annotation Hair orientation annotation	50K	Synthetic, Reconstructed	-	-
Hair-VAE[69]	816	Real	-	Front view	2,318	Synthetic, Reconstructed	Strand	Multi, no kinky/braid
HairRGB-D[57]	20k+	Real	30	Unconstrained view	20k+	Real	Depth	Multi, no kinky/braid

Chapter 3

2D Hair Analysis

Our 2D hair analysis studies the attributes of hair directly from 2D hair images. It includes 2D hair strands extraction, hairstyles pattern recognition, and braid structure analysis.

3.1 2D Hair Guide Strand Extraction

A human head typically consists of a large volume of hair strands and each hair strand has a small diameter [87], thus it is very difficult to extract each single hair strand from hair images. By carefully analyzing the hair images, we notice that the adjacent hair strands tend to be alike. The hair strands are parallel within a certain hair region and this holds especially true for straight hairstyles. Thus, it is possible to extract some **guide strands** that can represent the shape and structure of hair [62, 88, 63, 14]. Then those guide strands can be used to generate more (children) strands by interpolation. Thus, 2D guide hair strand extraction is an essential and important step of hair analysis and later for 3D hair modeling.

3.1.1 System Overview

The overview of our 2D guide hair extraction system is shown in Figure 3.1. The system has two steps: (1) 2D hair orientation map generation and enhancement. (2) Hair strand segment tracing and connection.

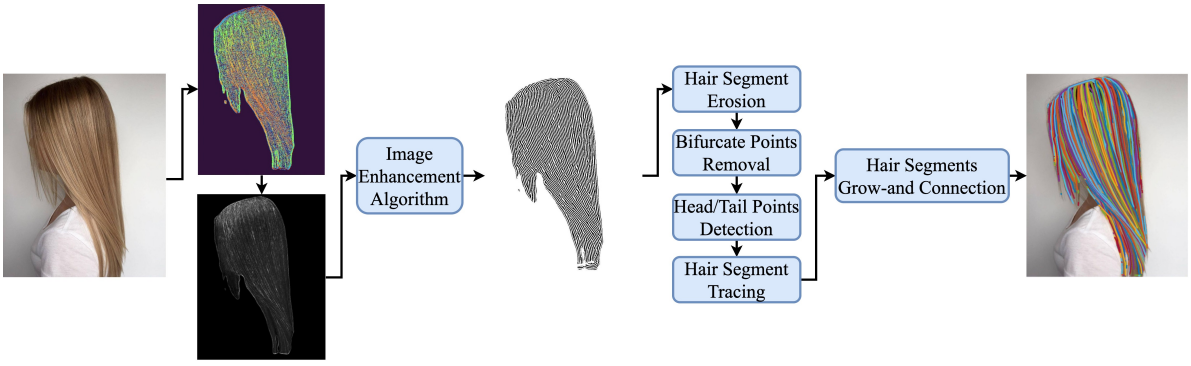


Figure 3.1: Overview of our 2D hair strand extraction method.

3.1.2 2D Hair Orientation Estimation

Paris et al. [62] first introduced dense orientation maps for hair modeling. Previous research works have shown that the Gabor filter is well suited to estimating the local orientation of hair strands. A 2D Gabor filter is constructed by modulating an oriented sinusoidal plane wave by a Gaussian envelope. The real component of an even-symmetric Gabor filter is given by [41].

$$G(u,v) = \exp\left(-\frac{1}{2}\left[\frac{\hat{u}^2}{{\sigma_u}^2} + \frac{\hat{v}^2}{{\sigma_v}^2}\right]\right)\cos\left(\frac{2\pi\hat{u}}{\lambda}\right) \tag{3.1}$$

where $\hat{u} = u\cos\theta + u\sin\theta$ and $\hat{v} = -v\sin\theta + v\cos\theta$. The λ and θ are the period and orientation of the sinusoidal plane wave. And the σ_u and σ_v determine the standard deviation of the Gaussian envelope.

Given a bank of Gabor filter G_θ generated by rotating an original x-aligned Gabor filter G_0 by angle $\theta \in [\,0, \pi)$, the orientation $\Theta(\,x, y)$ of image I at pixel $(\,x, y)$ is defined as:

$$\Theta(x,y) = \arg\max_{\theta} \left| G_\theta * I\,(x,y) \right| \tag{3.2}$$

In our system, we first apply the hair region segmentation method [51] to obtain the hair region. Then we convert the RGB hair images into HSV color space and we only keep the S (saturation) image and filter with a bank of Gabor filters to calculate the 2D hair orientation map.

In our experiments, we use the bank of Gabor filters with 10 different orientations from 0 degree to 180 degree, we also set $\sigma_u = 1.8$, $\sigma_v = 2.4$, and $\lambda = 4$. Based on a series of Gabor responses at each pixel position, the orientation that produces the high score response is stored in the orientation map and the maximum Gabor response is saved as the Gabor response image, as shown in Figure 3.2-(c).

By carefully analyzing the maximum Gabor response images in Figure 3.2-(c), we notice that within a local hair region: (1) the maximum Gabor response image keeps the appearance of the hair strands, (2) there is a pattern of alternated bright lines and dark lines. Thus we develop a hair strand extraction method inspired by the image enhance algorithm [32]. The enhanced algorithm can adaptively improve the clarity of hair strands in filtered hair images based on the local orientation and frequency. This image enhancement algorithm is computationally efficient and robust concerning the quality of input hair images.

3.1.3 Hair Strand Segments Connection

Given the enhanced hair images, we erode the lines in the image to give the hair strands a pixel-width presentation which is easy to track by using a standard line-tracing algorithm. However, the eroded image contains some individual points and bifurcation points need to be removed. We further remove the segments which are shorter than the predefined length threshold. Moreover, we use a spline with 25 control points to represent each hair strand segment. (The number of control points can be easily changed). However, the resulting hair segments are relatively short. To connect them into long hair strands, we apply a grow-and-connect technique. The procedure is shown as follows:

Step 1: Select segment c_i from the current segment set and grows from one of its ends with a predefined distance (controlled by the extension parameter ext_{para})of pixels along the tangent directions as $\widehat{c_n}$.

Step 2: Calculate the distance between the growing end of segment $\widehat{c_i}$ and other segments $c_{j,j\neq i}$.

Step 3: Out of the pair of segments $\widehat{c_i}, c_j$, choose the pair ($\widehat{c_i}$, c_k) that has the minimum distance.

Step 4: If the minimum distance is smaller than the predefined threshold (10 pixels) then connect the segments $\widehat{c_i}$ and c_k into c_{new}, and remove $\widehat{c_i}$ and c_k from the segments and add c_{new} to the hair segments. Save c_{new} in a new segment list. Otherwise, save $\widehat{c_i}$ as c_{new} in the resulting segment list.

Step 5: Repeat Step 1 to Step 4 on another segment $c_p \neq c_{new}$ in current segment set until all possible connections have been made.

Step 6: Increase the growing distance $d = 1.2 \times d$, and repeat step 1 to step 6 on the resulting segment set to predefined times (10).

After the hair segments are connected, we notice that some short segments have not been connected. Thus we remove them if they are shorter than the predefined threshold.

3.1.4 Experimental Results

Our 2D hair strand extraction results are shown in Figure 3.2. In our experiments, the input image is resized to 500 pixels$\times$ 500 pixels. In the hair segments connection stage, we set the spline extension parameter as $ext_{para} = 0.5$, the distance threshold for the connection is set as 5 pixel, the incremental parameters of the threshold of connection is l pixel, and the times of extension is 15. From Figure 3.2, we can see that our method can extract long hair strands from hair images.

A visual comparison with previous methods is shown in Figure 3.3. The extracted hair strands in [13, 14] are shorter than ours. Furthermore, both of the previous methods

Figure 3.2: 2D hair extraction results. (a) Hair images. (b) Orientation maps. (c) Maximum Gabor response images. (d) Enhanced hair strand images. (e) 2D hair strand extraction results. The input images of last two rows are from [13, 14].

require user's inputs. [13] needs manual strokes to indicate the hair region segmentation and hair growth direction and [14] requires the users to draw strokes to segment the hair region, face, body, and background. On the contrary, our method can extract long hair strands directly from hair images automatically.

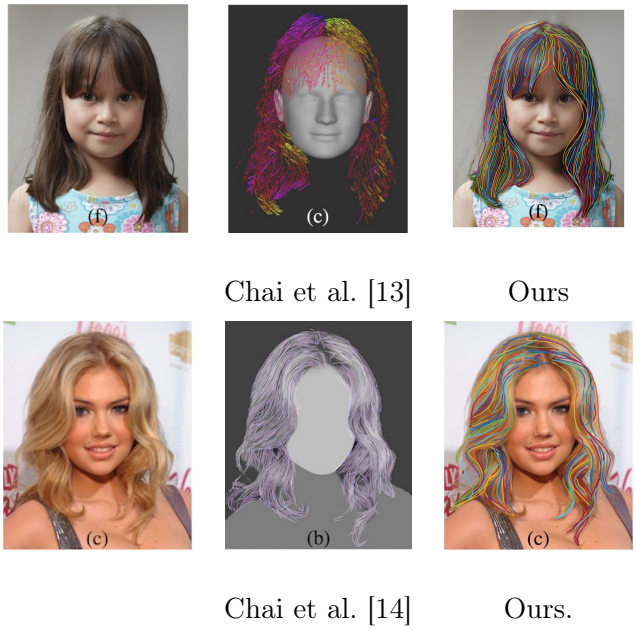


Figure 3.3: 2D hair strand extraction comparison.

Limitation Existing orientation-field based 2D hair strands extraction systems can only obtain straight and curly hair strands. We have tested our hair strand extraction method on the kinky hairstyles, as shown in Figure 3.4. It is not easy for a human to determine the strands from the kinky hair image (we have manually drawn a few strokes to represent the kinky hair strands, as shown in Figure 3.4-(d)), thus it is very difficult for orientation-based methods to extract kinky hairs.

The methods based on hair image orientation fields not only fail to extract the correct strand structures of kinky hairs but also do not work for braid structures. As shown in Figure 3.5-(b), the orientation field of kinky hair is very noisy, and the growth direction

(a) (b) (c)

Figure 3.4: A failure case for kinky hair strands extraction. (a) Input image. (b) The extracted 2D hair strands. (c) A few strokes (in green) that drawn manually to show the hair strands.

of hair strands can not be determined. In Figure 3.5-(d), the orientation fields along can not be used to obtain the overlapping structures between multiple braid strands. And this challenging problem is one of our main motivations for our 2D hair analysis work, we will revisit this part in Section 3.2 and Section 3.2.4 and provide a feasible solution.

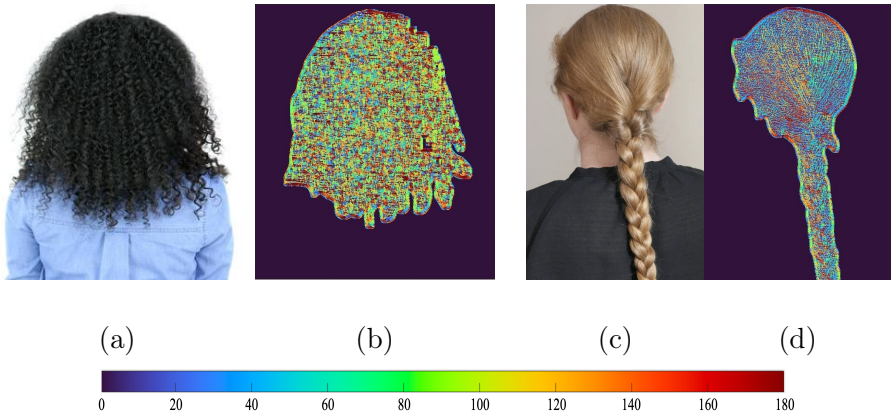


Figure 3.5: The orientation maps fails to represent the hair strand structures of kinky hairs and braids. (a) and (c) are input images. (b) and (d) are the corresponding hair orientation fields.

3.2 2D Hairstyle Pattern Recognition

Hairstyle is a crucial part of a person's appearance since it helps to provide certain identification and personality. Moreover, hairstyle is a supplementary feature for human recognition tasks [95, 17]. However, hairstyle recognition remains one of the most challenging tasks due to the unique characteristics of hairs, such as omnipresent occlusion, specular appearance, complex discontinuities, etc. Furthermore, for certain hairstyles, such as kinky hairs and braids, the hair strand information is difficult to obtain directly from images using traditional hair analysis methods. For example, as mentioned in Section 3.1.4, the kinky hair strands are nearly impossible to discover due to the tiny coiled structures and (usually) the dark color.

Real-world hairstyles span a diverse range of appearances. Existing hairstyle recognition systems focus on discovering the global information and overall structure, such as the shape of hair regions, the color of hair, the length of hair, and the orientation of hair strands. However, those systems fail to provide detailed information about the hair, such as the hairstyle patterns: straight, curly, kinky, braids, etc. Also, if the hair image contains several hairstyles, there is no information about the partition of those hairstyles within the hair region. Thus in this section, we will tackle those problems and provide the structural information of hair strands by leveraging the power of deep learning.

Given the complexity of hairstyles, we first provide the definitions of basic hairstyle patterns we used in our system:

-**Straight hairstyle** hair is normally straight and does not hold a curl

-**Curly hairstyle** hair contains spirals or inwardly curved forms or has a definite

'S' pattern

-**Kinky hairstyle** hair is tightly coiled with a less visible curl pattern, it is also called Afro-textured hairs

-**Braid hairstyle** two to five (can be more) groups of hair strands interlacing with each other to form a complex structure or pattern.

Based on how many hairstyle patterns appear in the hair region, we can further separate hairstyles into two types as follows:

- **The simple hairstyles** only contain a single hairstyle pattern within the hair region, as shown in the first row of Figure 3.6. However, even the simple hairstyles can be divided into dozens of subcategories based on certain criteria (e.g. the curly hairstyle can be divided based on the curliness).

- **The complex hairstyles** are the combination of two or more simple hairstyle patterns. Different combinations or distributions lead to various hairstyles, as shown in the second row of Figure 3.6.

Extracting the unique feature of each hairstyle pattern is crucial in hairstyle recognition. However, it is very difficult to use traditional image processing methods to obtain representative features for certain hairstyles. Thus, we apply the convolutional neural networks and leverage their strength to extract hairstyle features and perform hairstyle recognition. Our work is the first work to explore the hairstyle patterns and the corresponding partition within the hair regions by applying deep learning techniques.

Since each basic hairstyle is composed of certain repeated patterns according to their definitions, it is reasonable to start by recognizing these most basic hairstyle patterns

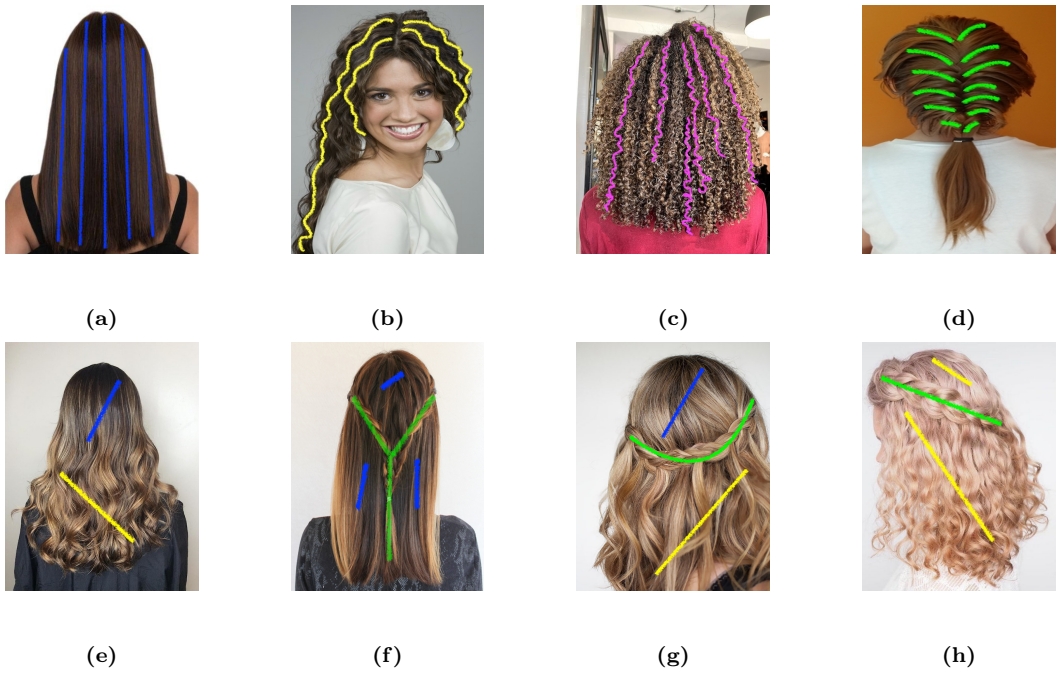


Figure 3.6: Hairstyles that contain a variety of patterns. The first row shows the simple hairstyles images. It includes: (a) Straight hair, (b) curly hair, (c) kinky hair, and (d) braid hair. The second row is complex hairstyle images. They are (e) Straight hair and curly hair combined hairstyle. (f) straight hair and braid combined hairstyle. (g) straight hair, braid, and curly hair combined hairstyle. (h) curly hair and braid combined hairstyle. (blue strokes: straight hair, green strokes: braid, yellow strokes: curly hair. pink strokes: kinky hair.)

within the hair region, then study the attributes and distribution of each hairstyle. This strategy improves the flexibility of our method when dealing with complex hairstyles. Thus, we design our system to train on hairstyle image patches that contain the distinct pattern, then to perform recognition on full hair images. The system has two parts: image-patch based hairstyle pattern recognition and full-image hairstyle pattern recognition. The image-patch hairstyle pattern recognition (in the green box) plays a fundamental role in our system. Since traditional image processing methods fail to extract the unique features of different hairstyles, we leverage the power of deep-learning methods to recognize the structure of different hairstyles. We trained our image-patch

based hairstyle pattern recognition network based on our hairstyle pattern image patch dataset. The network can recognize the straight, curly, kinky, and braid hairstyle patterns. When recognizing full hair images (in red box), we first divided the hair region into small patches, then each patch is sent to the network and provided a label, then we gather all the results of each patch to obtain the final recognition result for the full hair image. The recognition results can show the hairstyles as well as the partition of different hairstyles within the hair region.

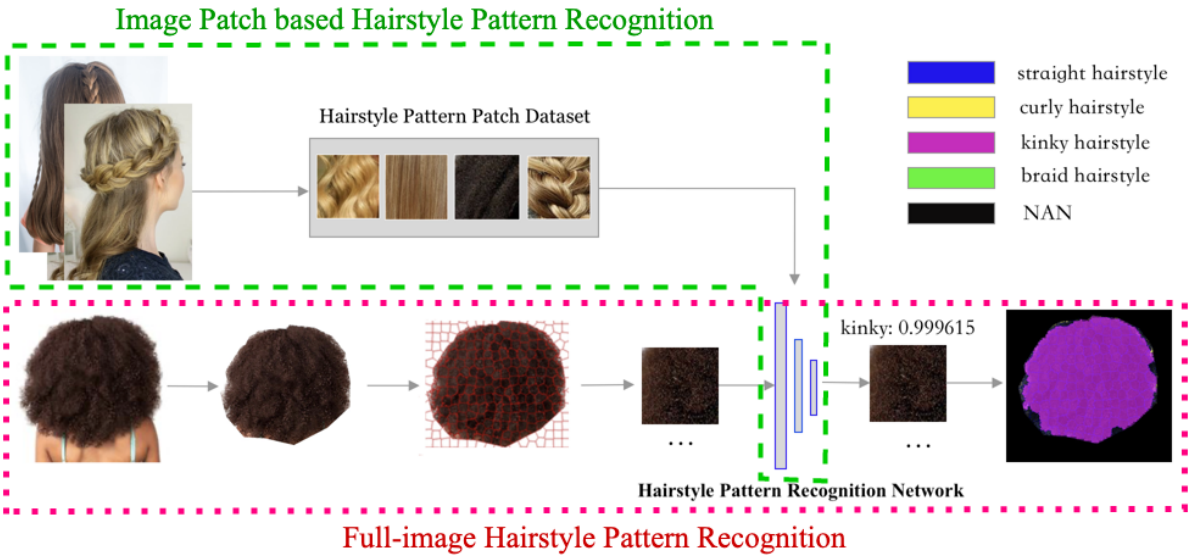
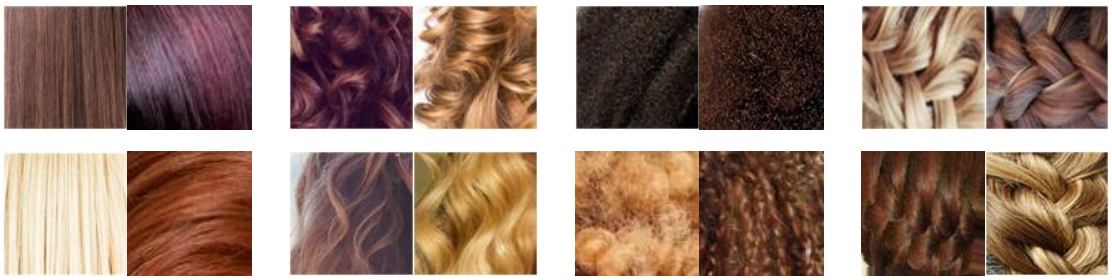


Figure 3.7: Overview of hairstyle pattern recognition system overview.

3.2.1 Hairstyle Pattern Dataset

Deep learning models require sufficient training data in order to have a good performance. However, there is no existing hairstyle pattern dataset available at the time when we develop the system, thus we create a hairstyle pattern patch dataset from scratch. All hairstyle images used in our system are obtained from the Internet (the license type is "creative commons licenses"). Those hairstyle images are captured from multi-views and

show a variety of hair colors, lengths, and volumes, etc.



(a) (b) (c) (d)

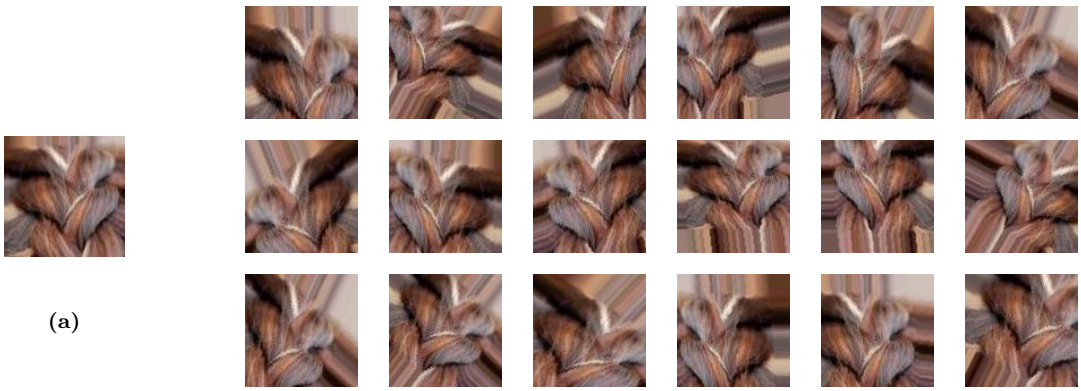
Figure 3.8: Image patches of four basic hairstyle patterns. They are (a) straight hairstyle patches, (b) curly hairstyle patches, (c) kinky hairstyle patches, and (d) braid hairstyle patches.

We have collected 253 images in total. The hairstyle images are separated into two groups: the training dataset (which contains 200 images), and the test dataset (which contains 53 images). In each group, the hairstyle images are evenly sampled. There are no overlapping images between those two groups.

In order to prepare the hairstyle patches training dataset, we manually crop the hair image patch within the hair region. Each hair image patch only contains one hairstyle pattern and is assigned with the corresponding label (straight, curly, kinky, and braid). The criterion of cropping is that we need to reserve the distinguish structures of different hairstyles. Take braid hairstyle, for example, if the cropping windows are very small, the image patches will lose the ability to represent their unique interlacing structure pattern and every image patch will look like a straight hairstyle. On the contrary, if the cropping window is very large, it may contain several hairstyles inside it and make the recognition difficult. Thus, instead of using a fixed-size window for hairstyle image patch cropping, we made the size of the cropping window adjustable and can capture the unique hairstyle structure of its kind. After the cropping procedure, we adjust the size of each hairstyle

image patch into 75 pixels × 75 pixels. Some examples of hair image patches are shown in Figure 3.8.

Then we separate all the hairstyle patches into the training dataset and testing dataset. For each hairstyle pattern, we use 1000 hair image patches for training, 300 hair image patches for validation, and 200 hair image patches for testing. Given our small training dataset, augmentation methods are used to increase the number of images and help to avoid over-fitting. Those methods include rotation, vertical shift, horizontal shift, shearing transformation, and horizontal flip. The data argument results for a braid image patch are shown in Figure 3.9.



(a)

(b)

Figure 3.9: Hairstyle image patch augmentation results. (a) The original braid patch. (b) Augmented braid image patches using rotation, vertical shift, horizontal shift, shearing transformation, and horizontal flip.

3.2.2 Image-patch Hairstyle Pattern Recognition

Given our limited data, we apply transfer learning for our hairstyle pattern recognition system. We take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset. We use Inception V3 network [76]

with a final layer retrained to our hairstyle pattern image patches dataset. The original Inception V3 network is trained on ImageNet [68]. We add a final layer retrained to our hairstyle dataset to learn features for different hairstyles. The hairstyle patch CNN model runs 1000 training steps. Each step chooses 10 hairstyle patches at random from the training set to predict the labels. The learning rate is 0.001. The loss function we used is the cross-entropy loss.

We run the testing procedure based on a set of hairstyle patches that are kept separate from the training and validation patches. The final testing accuracy researches 91.31%. The confusion matrix is shown in Figure 3.10. The recognition results are shown in Figure 3.11. From Figure 3.11, we can see that our image-patch based hairstyle pattern recognition network can recognize four basic hairstyle patterns (straight, curly, kinky, and braided hairs). In addition to the predicted labels, the prediction scores show the probability distribution of the reference hairstyle among the four basic hairstyles. Take the third image patch in the third row for example, the predicted label is the braid. The label with the second high score is curly. We can observe that this hair patch does contain the braid structure (lower part of the image) as well as some curly hairs (upper part of the image).

Some of the wrong recognition results are shown in Figure 3.12. Most of them are between the curly hair and braids since they have very similar structures in image patches.

3.2.3 Full-Image Hairstyle Recognition

The full images used in hairstyle recognition are selected from the test dataset described in 3.2.1. Since those images contain both hair regions and non-hair regions (e.g. faces, backgrounds, etc), the non-hair region should be removed before the hairstyle recognition

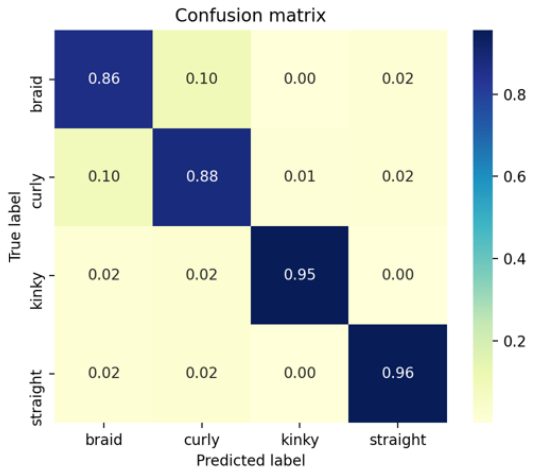


Figure 3.10: Evaluation on hair pattern patch dataset.

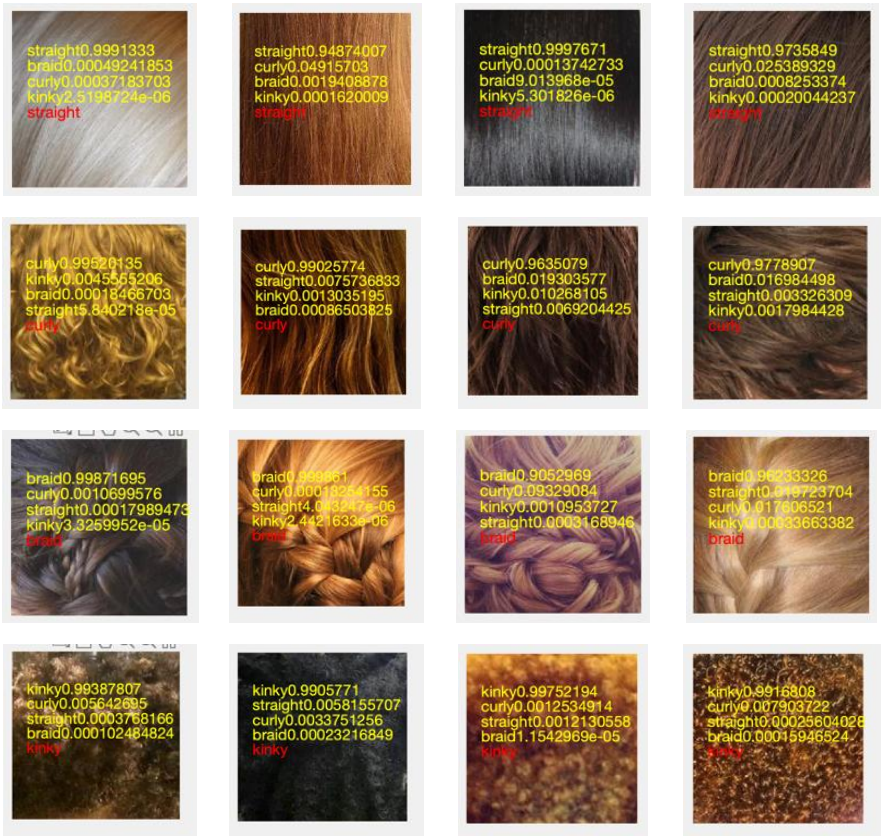
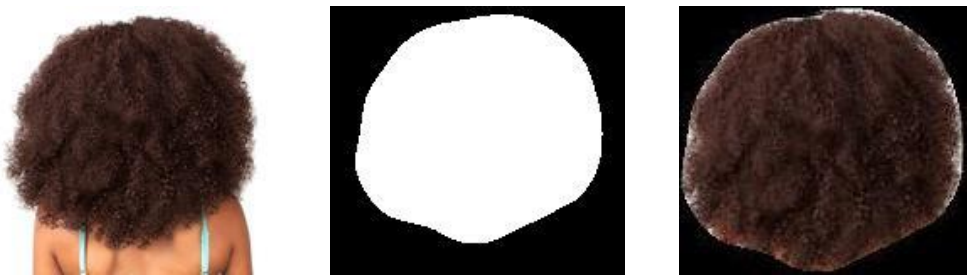


Figure 3.11: Patch-based hairstyle pattern recognition results. The yellow texts are the recognition results as (label, score), and the ground-truth label is red.

Figure 3.12: Failure cases of hairstyle pattern recognition.

procedure. We first apply the hair region segmentation method [51] to obtain the hair region, as shown in Figure 3.13. Furthermore, based on the hair region mask, we create a tight bounding box around the hair region and remove more background.



(a) **(b)** **(c)**

Figure 3.13: Hair region segmentation result. (a) Input hair image. (b) Hair mask. (c) Segmented and cropped hair region.

To conduct full-image hairstyle recognition, we need to first divide the hair region into sub-regions and perform our patch-based hairstyle recognition on each sub-region. For hairstyle images, there are no strong and clear boundaries between each basic hairstyle region. We need to find the rough boundaries between the hairstyle regions to perform recognition. Thus, the segmentation method we choose needs to satisfy the following demands:

- the algorithm should utilize the rich features of hair images
- the results should emphasize potential boundaries between different hairstyles, epi-

cally for braids

- the computational space and time should be reasonable

We divide the hair region into super-pixels by applying the simple linear iterative clustering (SLIC) algorithm [2]. The SLIC algorithm adapts a k-means clustering approach to efficiently generate superpixels. This algorithm generates superpixels by clustering pixels based on their color similarity and proximity in the image. This is done in a five dimension space, the first three dimensions are l, a, b , which is the pixel color vector in CIELAB color space. The remaining two dimensions are the pixel position x and y. The algorithm is described in Algorithm 1.

The SLIC algorithm is very simple to implement. Moreover, it is both time and memory-efficient. The superpixels generated by SLIC are compact, uniform in size, and adhere well to region boundaries.

The parameters control the superpixel segmentation procedure: k indicates the desired number of superpixels. The compactness parameters c of the SLIC algorithm control the shape of superpixels. A higher value makes superpixel more regularly shaped. A lower value makes superpixels adhere to the boundaries better. A number of iterations are used in the clustering phase of the algorithm, specified as a positive integer. For most problems, it is not necessary to adjust this parameter [2]. The superpixel segmentation results are shown in Figure 3.14.

Given the superpixel segmentation results, we need to generate hairstyle patches that satisfy the input requirements of the hairstyle pattern recognition network. Thus we

Algorithm 1: The SLIC superpixel algorithm [2]

Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.

Move cluster center to the lowest gradient position in a 3×3 neighborhood.

Set label $li = -1$ for each pixel i.

Set distance $d(i) = \infty$ for each pixel i.

repeat

  for *each cluster center* C_k **do**

    for *each pixel* i *in a* $2S \times 2S$ *region around* C_k **do**

      Compute the distance D between C_k and i.

      if $D < d()i)$ **then**

        set $d(i) = D$

        set $l(i) = k$

  Compute new cluster centers.

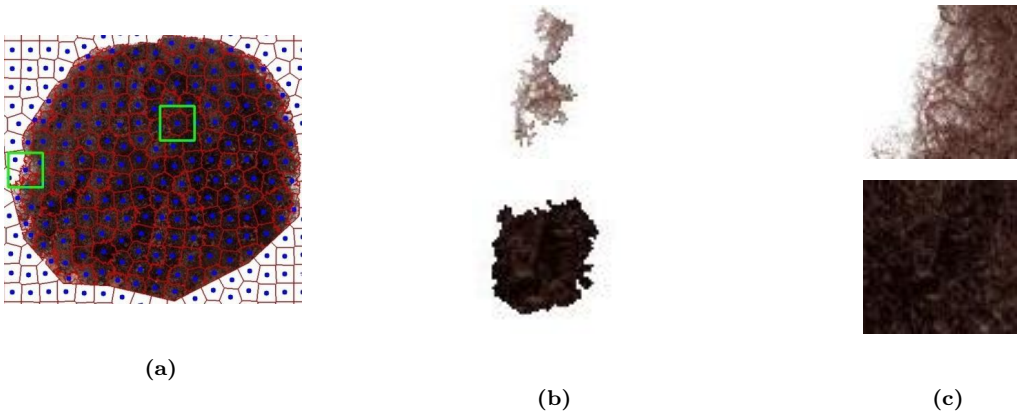
  Compute residual error E.

until $E \leq$*threshold.*;

calculate the centroid point of each superpixel using the following Equations 3.3:

$$X_i = \frac{\sum_{n=1}^{m} X_{in}}{m}$$
$$Y_i = \frac{\sum_{n=1}^{m} Y_{in}}{m} \tag{3.3}$$

where X_i and Y_i are the coordinates of the centroid point of the ith superpixel and m is the number of pixels in the ith superpixel. Regardless of the shape of the superpixel, we wrap all the pixels in a bounding box around each centroid point. Noting that the image patch should be larger than the superpixel patch. The generated image patches are shown in Figure 3.14-(c). As shown in Figure 3.14-(b), the upper superpixel lies on the boundary of the hair region, so the information contained in the superpixel is limited. However, the corresponding hairstyle patch can include more hairstyle information.



(a) **(b)** **(c)**

Figure 3.14: Hairstyle patch generation results. (a) Superpixel segmentation. The red lines indicate the boundaries of the superpixels. The blue dots are the centroid points of the superpixels. The green boxes indicate the hairstyle image patches that are generated based on the centroid points. (b) Two hair superpixels. (c) The corresponding hair patches.

For each hairstyle image patch p_i , the hairstyle recognition result is composed of the class labels and the corresponding scores $(label_n, score_n)$. Noting that the scores satisfy $\sum_{n=1}^{4} score_n = 1$. If the score of the $i-th$ image patch is larger than the predefined threshold value $threshold(= 0.75)$, then the corresponding label $label_n$ will be assigned

to the $i-th$ image patch as well as the corresponding $i-th$ superpixel. For example, the label and score of the above two hair patches are shown in Table 3.1. From Table 3.1, we can see that image patch No. 2 is correctly recognized as the kinky hairstyle. However, hair image patch No. 1 can not be recognized as any of the hairstyle patterns because of a lack of information. The recognition result for the complete hairstyle image is shown in Figure 3.15. Based on the recognition results, we calculate the number of pixels of each hairstyle and compare it with the total number of pixels of the hair region. Then we obtain the dominant hairstyle of the input image. The dominant hairstyle of Figure 3.15 is considered the kinky hairstyle since the kinky hair pixels are about 99.9% of all the hair pixels.

Table 3.1: Hair image patch recognition results.

Hair image patch No 1		**Hair image patch No 2**	
Label	**Score**	**Label**	**Score**
straight	0.000211166	straight	1.00396e-05
curly	0.574778092	curly	0.000369855
kinky	0.420464	kinky	0.999615
braid	0.00454679	braid	5.06906e-06

We conduct experiments on different hairstyle images, including both simple hairstyle images and complex hairstyle images captured from multi-views.

Figure 3.16 shows the hairstyle pattern recognition results on straight, curly, and kinky hairstyles. As we can see, in the second row of Figure 3.16, there are several hairstyle patches that are recognized as the kinky hairstyle with high scores. However, by careful observation, we found that those hairstyle patches actually belong to the

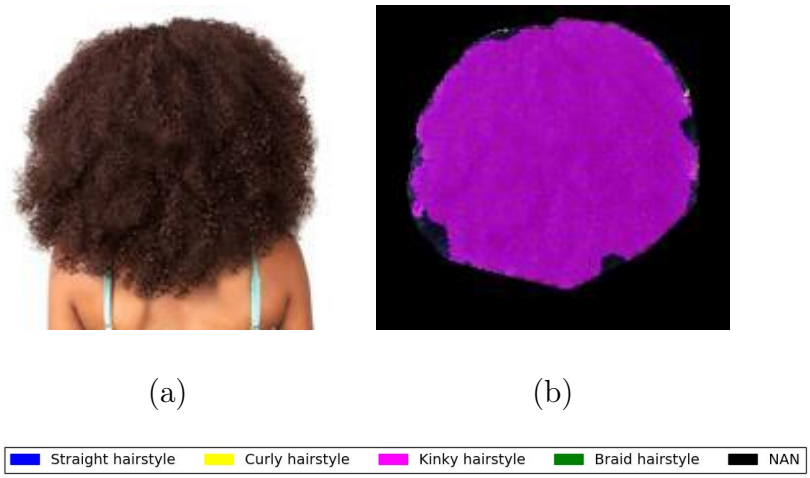


Figure 3.15: Hairstyle Recognition Result. (a) The input hair image. (b) The hairstyle is recognized as kinky hairstyle for the whole hair region.

kinky hairstyle. The recognition results indicate that our system can recognize the kinky hairstyle.

The recognition results of complex hairstyles are shown in Figure 3.17. The first row shows the combination of straight hairstyle and braid hairstyle. We notice that there is one 'braid' patch at the top of the hairstyle image and a 'straight' patch at the tail of the braid. The 'braid' patch is very near the dividing line in the hair region. Since the appearance of this region shows an interlacing structure, the hairstyle patch is considered to be the braided hairstyle. The 'straight' patch only contains a limited hair region and is very similar to the straight hairstyle. It is worth mentioning that there is a clear dividing line between the straight hairstyle region and the braid region. The hair image of the second row in Figure 3.17 can be described as very loose and contains the "S" pattern. From the recognition results, we can find that the upper part of the hair region is recognized as the straight hairstyle and the lower part is recognized as the curly hairstyle. The recognition results of a more complex braid structure combined with the

straight hairstyle are shown in the last row of Figure 3.17. Although there is two braid interlace with each other, our method can recognize both of them.

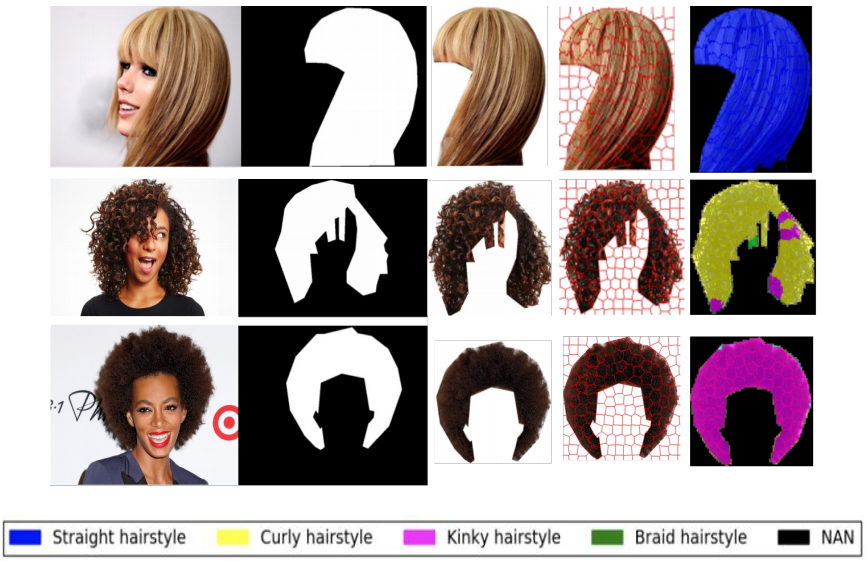


Figure 3.16: Simple hairstyle recognition results.

Figure 3.17: Complex hairstyles recognition results.

Ablation Study In this part, we study how the number of superpixels, as well as the size of the image patch, will affect the performance of our full image recognition system.

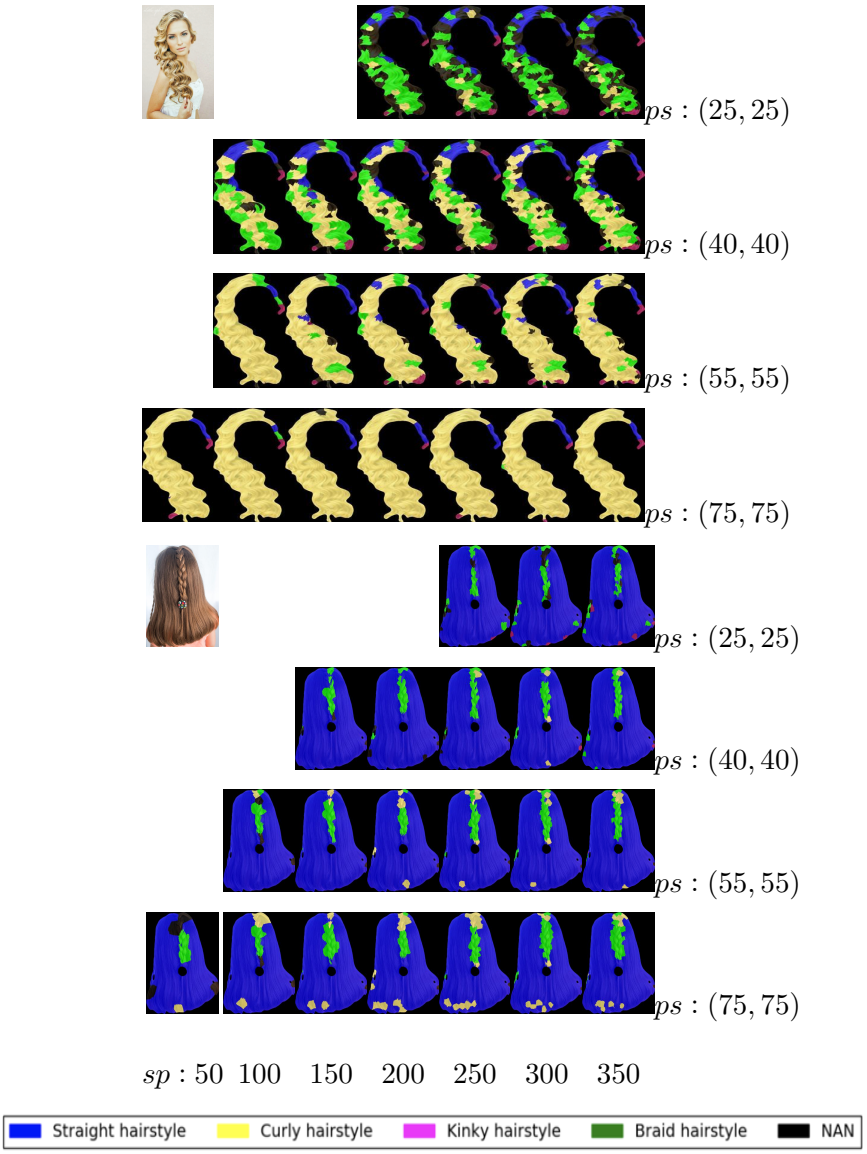


Figure 3.18: Comparison between different number of super pixels and different sizes of image patch. ps: patch size, sp: number of super pixels.

The number of superpixels represents the sample rate. A large number indicates the image is divided into more irregular grids. The small number indicates the sample locations are more sparse. Meanwhile, the patch size controls the amount of data extracted from the image. A large patch shows more hairstyle information but may also contain

two or more hairstyles at the same time. A small patch contains only one hairstyle, however, may limit the characteristics of different hairstyles pattern. As shown in Figure 3.10, the curly hair and braids have less satisfying recognition results, thus we use the curly hair and braids as examples.

The input image of our system is all resized proportionally to make sure that the larger dimension is no more than 600 pixels. The resolution of the curly hair region is 304×532 and the resolution of the braid hair region is 325×467. The number of superpixel we used in this experiments are $[50, 100, 150, 200, 250, 300, 350]$, the patch sizes are $[25 \times 25, 40 \times 40, 55 \times 55, 75 \times 75]$. The comparison results are shown in Figure 3.18. For curly hairstyles, a larger patch size leads to better recognition results. When the patch size reaches 75, the recognition results are relatively good. This indicates that if the hairstyle image is a simple hairstyle, a larger patch size leads to better results. For a simple hairstyle, a larger patch will contain the complete hairstyle pattern. On the contrary, when a small patch size is used, as shown in the first row of curly hair, most of the patches are considered braids. This is because this curly hair has very strong curves, when the curve changes its directions, it is very similar to the braid structure. For the braid hairstyles, there is no dramatic visual change in the results. However, by carefully examining the results, we find that when using the patch of 55×55, the recognition of the braid shows a more compact boundary. Thus in our experiments, we chose the number of the superpixel as 200 and the patch size as 65×65 in all our experiments.

3.2.4 Hairstyle Representation Analysis

In this section, we extend our investigation on the hairstyle pattern recognition results. We are trying to find the relation between hairstyle recognition results and the parametric

representation of hair strands.

We defined our four basic hairstyle patterns by analyzing Andre Walker's hair typing system [19]. The definitions of different hairstyles of Andre Walker's system are shown in Figure 3.19 and the corresponding exemplars are shown in Figure 3.20.

Type	Hair texture	Hair description
1a	Straight (fine)	Very soft, shiny, hard to hold a curl, hair tends to be oily, hard to damage.
1b	Straight (medium)	Has much body. (i.e. more volume, more full).
1c	Straight (coarse)	Hard to curl (i.e. bone straight).
2a	Wavy (loose waves)	Can accomplish various styles. Loose "S" pattern. Hair sticks close to the head.
2b	Wavy (defined waves)	A bit resistant to styling. Hair has more of a defined "S" pattern. Hair tends to be frizzy.
2c	Wavy (wide waves)	Hair has wider waves. Resistant to styling. Hair tends to be frizzy.
3a	Curly (loose curls)	Thick and full with much body. Definite curl pattern. Hair tends to be frizzy. Can have a combination texture.
3b	Curly (tight curls)	Medium amount of space of the curls. Can have a combined texture.
3c	Curly (corkscrews)	Tight curls in corkscrews. The curls are very tightly curled.
4a	Kinky-coily (defined coil)	Tightly coiled. Has a very defined "o"-shaped pattern.
4b	Kinky-coily (z coil)	Tightly coiled. Little less defined kink pattern. Has more of a "Z"-shaped pattern.
4c	Kinky-coily (tight coil)	Tightly coiled. Almost no visible defined kink pattern, unless seen from up close. Has more of a very tight "o"-shaped pattern.

Figure 3.19: Andre Walker's hair typing system [19].

Figure 3.20: Exemplars of Andre Walker's hair typing system [19].

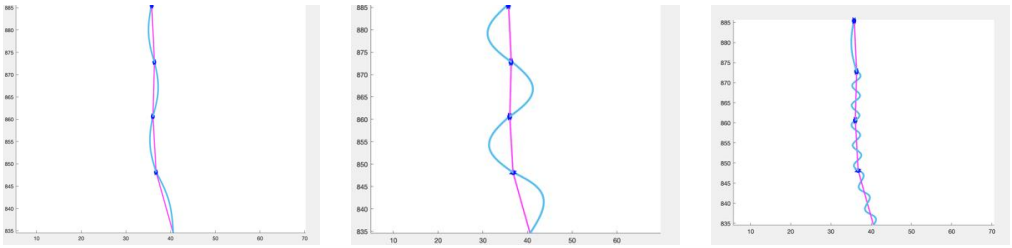
In Section 3.1, we represent the hair strands using spline with a predefined number of control points (about 20 in our experiments), which is consistent with the industry's standards [23]. And this can be easily extended to 3D.

We define a 3D hair strand as $\zeta = \{h_i\}_{i=0}^{M}$, where h_i is the hair strand segment. We use Helix function to represent each hair segment, as shown in Equation 3.4 .

$$\begin{aligned} t_i &= 0 : intvl : len_i \\ y_i &= r * sin(f * t_i) \\ z_i &= r * cos(f * t_i) \end{aligned} \tag{3.4}$$

where len_i is the distance between the two control points that define the current hair segment. $intvl$ controls the number of points of each segment (= 10 in our experiments), r and f control the amplitude and the frequency of the curves, and finally (t_i, y_i, z_i) represent the coordinates of the points of the $i-th$ hair segment.

Changing the parameters r and f will also change the shape of the corresponding 3D strand. And since each hair strand has several hair segments, by controlling the shape of each segment, the strand will have a variety of appearances, as shown in Figure 3.21.



(a) (b) (c)

Figure 3.21: Hair strand shapes controlled by $f$and r. (a) 3D hair strand with f=1, r=1. (b) 3D hair strand with r=5, f=1. (c) The first segment of the the 3D hair strand with r=1, f=1, and the rest with r=1, f=5. The pink lines are 3D hair segmentations. The blue points are the control points on 3D hair strands. The light blue curves are hair strand splines controlled by parameters f and r.

Based on the Andre Walker's hairstyle system and the parametric representation of our hair strands, we can conclude the following rules:

- For the straight hairstyle, there are barely any curves between each pair of control points. In other words, the number of curves should be one at most and the amplitude of the curves should be very small.

- For the curly hairstyle, there is a median number of curves between each pair of control points. And the amplitude of the curves should very also be median. This will lead to the 'S' shape of the curly hair.

- For the kinky hairstyle, there are more curves between each pair of control points and the amplitude of the curves should be small.

In section 3.2.3, we have conducted the hairstyle pattern recognition, and the results show that we can successfully recognize the hairstyle information and its corresponding distribution within the hair region. By combining the hairstyle recognition results, the extracted hair strands(2D or 3D), and the above rules, we can control and adjust the shape of our generated hair strands. However, it's very difficult to get the r and f information directly from the hair image, especially for the kinky hairstyle.

One possible way to change the frequency f according to the hairstyle pattern could be, firstly we normalize the hair strands (the hair has the dimension of $130pixel \times 130pixel$), make each hair segment has a fixed number of curves with a fixed amplitude, then change the number of hair segments along each hair strand. We have conducted experiments to obtain different hairstyles by changing the number of segments and the results are shown in Figure 3.22. From Figure 3.22-(a) to (e), the hairstyle changes from straight hair to curly hair, then to kinky hair. Especially, Figure 3.22-(d) to (f) can cover most of the common kinky hairs.

For the amplitude of the curves, we have also conducted experiments, as shown in

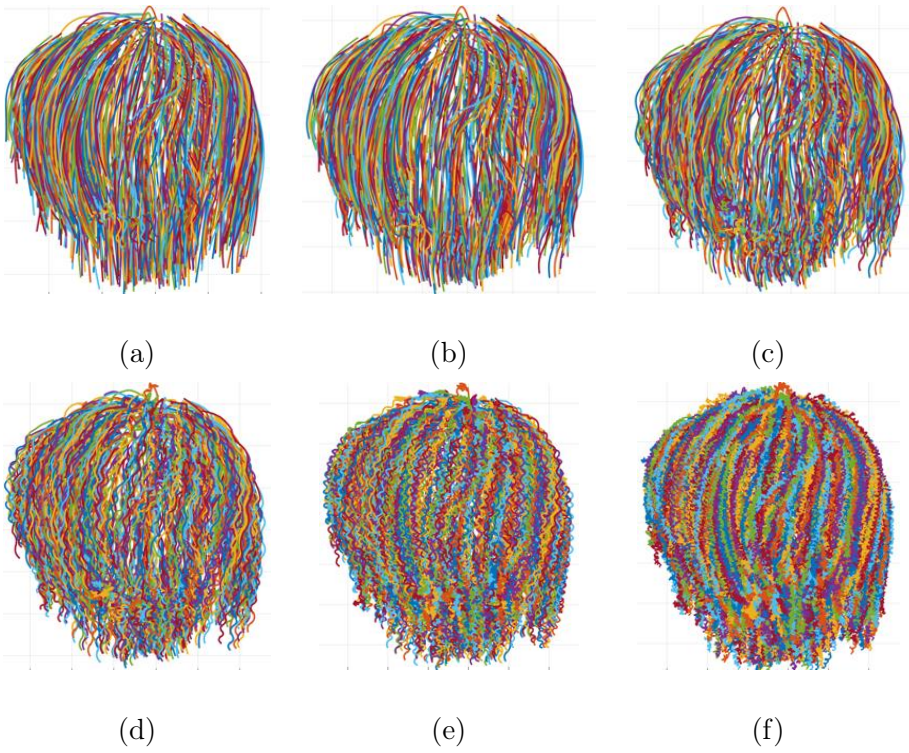


Figure 3.22: Hair strand shape controlled by the number of hair segments. Each hair strand in this hair model has 100 points. (a) the original hair strands. From (b) to (f), each hair strand has 3, 5, 10, 20, 50 hair segments, correspondingly.

Figure 3.23. Based on our observations of the real world hairstyles, and by considering Andre Walker's hairstyle system, we feel that most of the curly hairs are in the range between Figure 3.23-(b) to (d), and maybe on a very rare occasion in Figure 3.23-(f).

The above analysis is mainly based on comparing the visual appearance, the extraction of the curliness of curly hair and coil structure of the kinky hairstyles is not easy. However, we have estimated a reasonable range for parameter selection as shown in Table 3.2, which will help us recover the curly and kinky hairstyles.

Future exploration on extracting the parameters may be achieved by combining deep-learning models with our parameter-controlled kinky and curly hair models. Our work provides a high potential for a bright future in curly and kinky hair modeling.

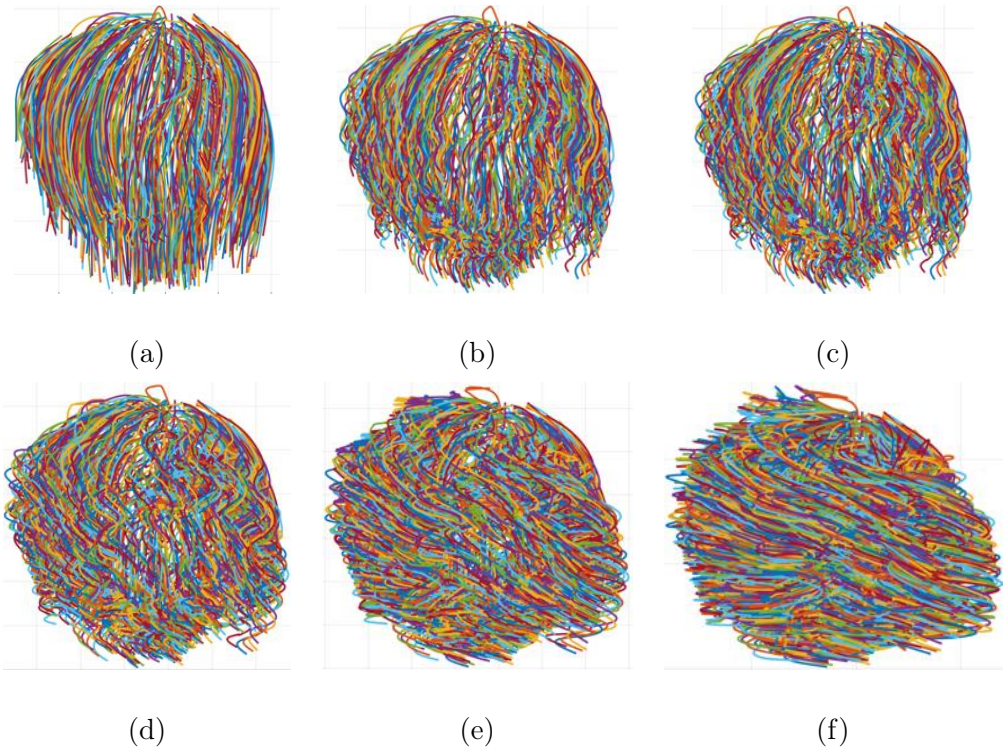


Figure 3.23: Hair strand shape controlled by amplitude of the curve. Each hair strand in this hair model has 100 points and has been evenly divided into 5 segments. (a) the original hair strands. From (b) to (f), the amplitude is 1, 2, 5, 10, 20 (pixels), correspondingly.

Table 3.2: Representative parameters of different hairstyles. Ampli.: Amplitude.

	straight hair	curly hair	kinky hair
No. of segments	3-5	5-10	15-20
Ampli. of curves	0	2-5	1-2

3.3 Braid Structure Analysis

Existing 2D hair strand structural analyses become problematic for constrained hairstyles such as braids since their priors cannot properly model the intertwined topologies of braids. [37] is the only recent paper that talked about 3D braid reconstruction. Their

system first calculates the local 3D orientation fields based on captured braid points cloud, then the braid point cloud is aligned with the patches in the predefined 3D braid patch datasets based on random sampling fitting. Users' interactions are required to choose a similar braid model from the dataset or to provide a rough initial scale to help speed up the searching. At last, the optimal braid patches are calculated and the center lines of the braid patches are connected to generate the final complete braids. Although it provides a solution to recover the braid structure, however, this method suffers from the following disadvantages:

- A capture system is required to obtain the 3D braid point clouds.

- A predefined 3D braid patch dataset needs to contain a variety of 3D braid patches.

- User interaction is required to speed up the searching procedure of braid candidates

- Since the input of their system must be the braid point clouds, no matter how these point clouds are generated, it requires a real object being scanned, or multi-view images being captured. Thus, if we want to recover the braid structure in any single-view image, their system will fail.

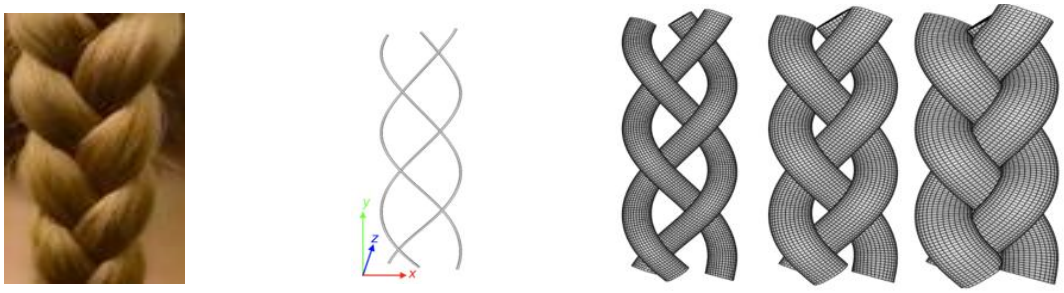
In our work, we discover the 2D braid structure from a single-view braided hair image, leveraging the power of deep learning models.

A braid can be represented by several centerlines $S_i, i = 2, ...n$. Take a typical three-strand braid for example, the structure can be described using Equation 3.5 and the

corresponding 3D braid patch is shown in Figure 3.24.

$$\begin{aligned} \mathbf{S_0} &: x = a\sin(t), && y = t, b = b\sin(2t) \\ \mathbf{S_1} &: x = a\sin(t + 2\pi/3), && y = t, z = bsin(2(t + 2\pi/3)) \\ \mathbf{S_2} &: x = a\sin(t + 4\pi/3), && y = t, z = bsin(2(t + 4\pi/3)) \end{aligned} \tag{3.5}$$

where the y is the braiding direction, and a and b are two constants that determine the shape of the braid. Other braids can be similarly modelled according to how the strands are woven. However, as shown in Figure 3.24-(c), even if the thickness of the braid can be changed by user, the looks of the braids are not natural. Since in real life, the directions and thickness of the braid strands are changing based on how the braid is made and how many hair strands are gripped into the braid.



(a) **(b)** **(c)**

Figure 3.24: A three-strand braid exemplar[37]. (a) A three-stand braid image patch. (b) The braid strand centerlines. (c) Gradually expanding patches.

Analysis of 2D braid structure based on braid unit identification from single-view hair image has not been studied. Here we first show our definitions of important parts of a 2D braid as follows and visualize them in Figure 3.25.

-**Braid Units** Braid units are the visible sections on the braid. They are the basic elements of the braided hairstyle. They may be variants on shapes, orientations,

and arrangements. They provide the details of the braid's appearance.

-Braid region guide curve Braid region guide curve shows the orientation and length of the braid. Each braid usually has only one braid region guide curve.

-Braid strand guide curve The amount of braid strand guide curves and how they are woven define the overall appearance of braids. For example, French braids, and Dutch braids are both three-strand braids. However, a Dutch braid is the reverse of a French braid.

(a) **(b)** **(c)**

Figure 3.25: Braid structure definitions. (a) The braid region guide curve is in yellow. (b) The braid strand guide curves are in red, blue, and pink). (c) The braid units. The braid units are circled by green ellipses, and the center points of the braid units are shown as white points.

Our 2D braid structure analysis aims to automatically extract the braid units, braid region guide curves, and braid strand guide curves from 2D braided hair images. The overview of our system is shown in Figure 3.26.

Although the braid is composed of several hair strands, those hair strands are only partially visible in braid images. This is because the braid hair stands will overlap with each other, the strand under the overlapping part is covered by other strands. Thus, the appearance of the braid is made of a group of braid units.

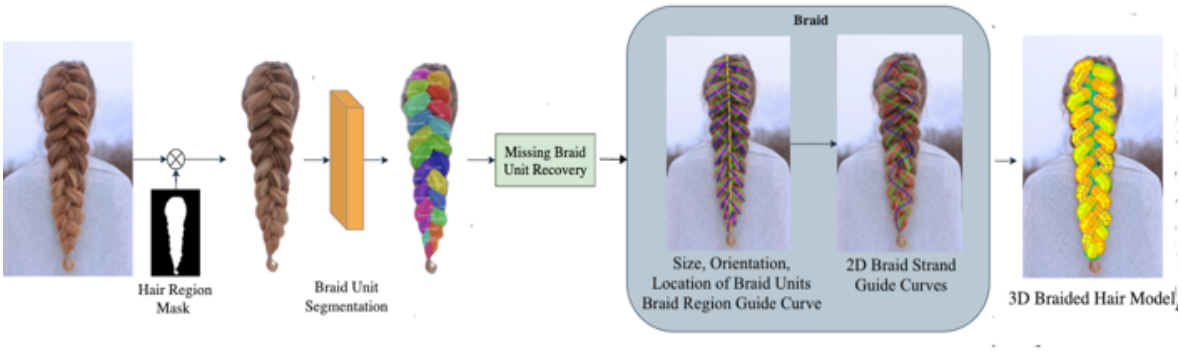


Figure 3.26: Overview of braid structure generation system.

Variance in the arrangement, shape, size, orientation, and locations of the braid units lead to different braid hairstyles. Thus as the basic elements of the braided hairstyle, the braid units can provide important information about the braid structure. In our proposed 2D braid analysis system, we treat each braid unit as an instance, then we apply instance segmentation methods to segment the braid units from 2D hair images.

Instance segmentation is the task of identifying object outlines at the pixel level. It is an important step to achieving comprehensive object detection and image recognition. Mask Regional Convolutional Neural Networks (Mask R-CNN) [31] perform segmentation of each object at the pixel level and separate each object from its background. The framework of Mask R-CNN has two stages: first, it scans the image to generate proposals, which are areas with a high likelihood to contain an object. Second, it classifies these proposals and generates bounding boxes and masks.

3.3.1 Braid Unit Dataset

Since there are no braid unit datasets available, we create our braid unit dataset. We download 225 braid hairstyle images from the Internet (the license type is "commercial

use and modification allowed"). Those braid hairstyle images contain different hair colors, braid lengths, strand amounts, and arrangements. Then we separate the images into two groups for training and testing respectively. We use the annotation tool VGG Image Annotator [31, 21] to manually annotate the braid units, as shown in Figure 3.27-(a). We use a set of polygon pints to represent each braid unit mask, as shown in Figure 3.27-(b), the braid units are annotated using yellow/orange polygons. Since our data set is relatively small, we applied data augmentation techniques to expand the training data set.

(a) **(b)**

Figure 3.27: Braid unit annotations. (a) The interface of VGG Image Annotator (VIA). (b) The braid units are annotated using (yellow/orange) polygons.

3.3.2 2D Braided Unit Segmentation

The overview of our braid unit segmentation and analysis system is shown in Figure 3.28. We apply transfer learning and use the weights that have been trained on the COCO dataset [53]. We implement the Mask R-CNN using the ResNet 101 and FPN backbone. During training, we set the learning rate at 0.001. We trained the network 100 steps per epoch for 50 epochs. The accuracy of the model is 98.72% and the precision is 96.94%.

The braid unit segmentation results are shown in Figure 3.29. After the braid unit

segmentation, we fit an ellipse to every braid unit contour. The center points of the ellipses indicate the locations of the braid units. We use the major and minor axes to represent the orientation of the braid unit.

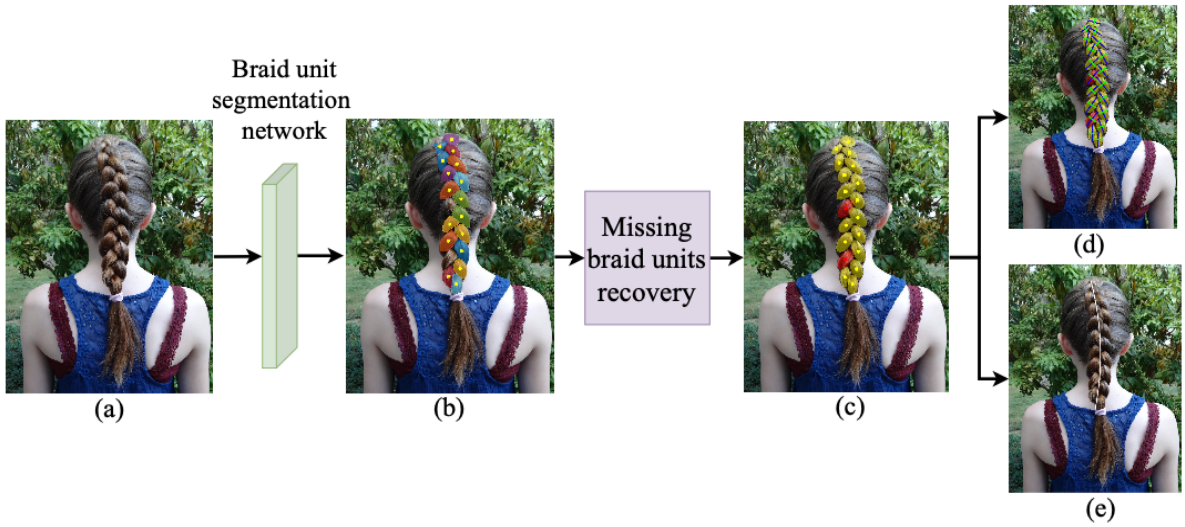


Figure 3.28: Overview of our braid unit segmentation and analysis method. (a) The input braid image. (b) Braid units segmentation result. (c) Missing braid units recovery. (d) The locations, sizes, and orientations of braid units. (e) The braid region guide curve.

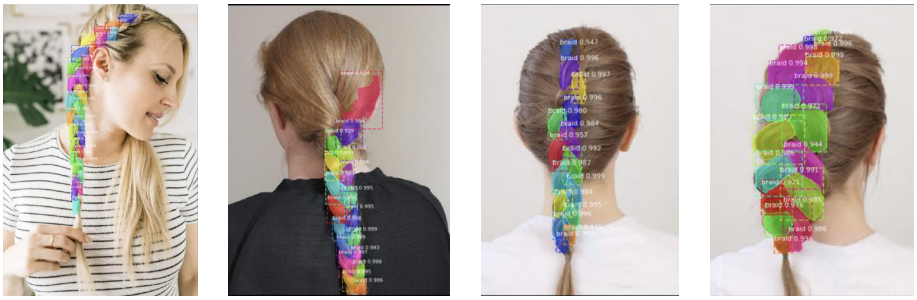


Figure 3.29: Braid units segmentation results. The braid units are color-coded.

3.3.3 2D Braid Structure Generation

Due to the complexity of the braid structure, there are still a few braid units that can not be detected, as shown in Figure 3.28-(b). Our approach missed two braid units on the left side of the braid region. Based on the center points of segmented braid units, we fit a curve using quadratic regression, then we can separate the segmented braid units into left units and right units. Then we calculate the indices where are missing units as well as the number of missing braid units following Algorithm 2.

In Algorithm 2, $munit\_idx$ is the list of indices in the original braid unit list where there are missing units, and $munit\_cnt$ is the list of a number of missing units for each item in $munit\_idx$. As shown in Figure 3.28-(c), $munit\_idx_1 = 4$ and $munit\_cnt_1 = 1$ mean that there is 1 missing braid unit after the 4th original braid unit. In the same manner, $munit\_idx_2 = 6$ and $munit\_cnt_2 = 1$ mean that there is 1 missing braid unit after the 6th original braid unit.

By analyzing braid images, we found that the sizes of the adjacent braid units decrease slightly towards the end of the braid and that the distance between braid units is also decreasing. Moreover, the orientations of adjacent braid units are similar. Thus, we use the information of adjacent braid units to estimate the location, size, and orientation of the missing braid units. Take the first missing braid unit in Figure 3.28-(b) for example, the center point of the missing braid unit is located in the middle of its adjacent braid units' center points. The major axis length, the minor axis lengths, and the orientation of the missing braid unit are calculated by averaging the major axis lengths, minor axis lengths, and orientations of adjacent braid units, correspondingly. After recovering all the missing braid units, we estimate the braid region guide curve by quadratic regression

Algorithm 2: Calculate missing braid units. We use this algorithm once for the left units and once more for the right units.

Input : lists of braid unit center points on the left and right sides

cpts_l$_i, i \in \{1, \ldots, m\}$ and cpts_r$_j, j \in \{1, \ldots, n\}$;

the average distances of dist_l and dist_r correspondingly, and set avgDist as the smaller average distance between those two;

distance ratio threshold th = 1.4;

Output: list of indices in the original braid unit list where there are missing units

munit_idx;

list of number of missing units for each item in munit_idx$_N$ munit_cnt ;

Calculate the distances between consecutive braid unit center points on the left side dist_l$_p, p \in \{1, \ldots, m-1\}$, and on the right side dist_r$_q, q \in \{1, \ldots, n-1\}$, correspondingly;

//Loop for the left side of the braid $N = 1$;

for $p \leftarrow 1$ *to* $m - 1$ **do**

    cnt1 $= floor(\frac{\text{dist\_l}_p}{\text{avrDist}})$;

    if $\frac{dist\_l_p}{avrDist} > th$ **then**

        munit_idx$_N = p$;

        if $p \neq m - 1$ & $\frac{dist\_l_{p+1}}{avrDist} < th$ **then**

            cnt2 $= floor\left(\frac{\text{dist\_l}_p}{\text{th} \times \text{dist\_l}_{p+1}}\right)$;

        else

            cnt2 $= floor\left(\frac{\text{dist\_l}_p}{\text{th} \times \text{dist\_l}_{p-1}}\right)$;

        $t = min\,(\text{cnt1}, \text{cnt2})$;

        munit_cnt$_N = \max(1, t)$;

        $N++$;

//We run a similar loop for the right side of the braid.

based on the center points of all the braid units, as shown in 3.28-(e).

A braid can have varying widths at different locations along the braid. The amplitude thus varies along the braid. We fit two quadratic curves to the left and right braid units and compare the distance to the braid region guide curve. Take a three-strand braid for example, the amplitude can be calculated from the braid region guide curve to either the left or right braid unit curves. We sample the braid region guide curve and braid unit curves with the same number of points and we set the varying amplitude as the distance between the pairs of corresponding points on the center and braid unit curves.

Then we need to group the braid units to their corresponding braid strands. Based on the braid region guide curve, the braid units can be divided into two groups: $l_i, (i = 1, ..., n)$ and $r_i, (j = 1, ..., m)$. As illustrated in Figure 3.30-(c), we choose consecutive 3 braid units from the left side $l_i, (i = 1, 2, 3)$ and corresponding 3 consecutive braid units from the right side $r_j, (j = 1, 2, 3)$. Based on the three-strand braid structure knowledge, we know that l_1 and r_3 belong to the braid strand guide curve 1, the l_2 and r_1 belong to braid strand guide curve 2, and l_3 and r_2 belong to braid strand guide curve 3. And this pattern repeats till the tail of the braid, as shown in Figure 3.30-(b).

Then we apply helix curves, which are similar to the technique used in [37], to approximate the braid strand guide curves, as shown in Figure 3.30-(c). Take a three-strand braid for example, it can be represented by three braid strand guide curves $curv_n (n = 0, 1, 2)$ using the following equations.

$$curv_n : \begin{cases} x = a \sin(t + n * \pi/3) \\ y = t \end{cases} \tag{3.6}$$

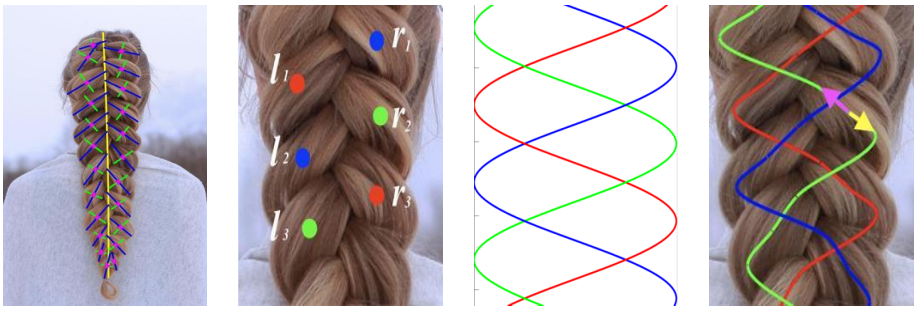
where (x, y) represents the coordinates of each point of the braid strand guide curves.

y is the braiding direction. a is related to the width of the braid unit, and b controls the thickness of the braid.

However, the above method only uses the information between two braid center curves, the other half (the part outside of the braid unit center curve) of the braid units is not considered. Thus, the generated braids may lack some of the details from the braid input image, especially if there are some irregular braid unit shapes.

To create a realistic braid structure, we proposed a braid structure refinement method that makes the most use of the information from the braid image. We first generate bilateral line segments along the major axis of the ellipse from the center point of each braid unit, as the yellow and magenta segments in Figure 3.30-(d). The length of each line segment is 1/3 of the long axis length of the current braid unit. Thus, the braid strand guide curves keep the orientation details of the braid units of the input braid images. Then we connect the line segments which belong to the same braid strand one by one. Since the connection parts are invisible in the braid image, we use straight lines to connect, as shown in Figure 3.30-(d).

Braids are naturally 3D objects. When several groups of braid strands overlap each other, the visible parts (braid units) will pop up, and the invisible parts hide under the braid units. In our system, we use 3D tubes to represent the 3D braid strand guide curves. The radius of the 3D tube is based on the length of the minor axis b. When the sections of the tube belong to the braid units, we pop it up by the depth of $\lambda * b$. For the invisible parts, we gradually change the depth from $\lambda * b$ to $-\lambda * b$ then back to $\lambda * b$, so the "invisible" parts can be smoothly connected with the braid units. Although the resulting braid structure is in 3D, it is more a "flat" braid rather than a true 3D braid. We use a "tube" to represent each braid strand guide curve for visualization purposes,



(a) (b) (c) (d)

Figure 3.30: Braid structure analysis. (a) The center points (magenta), major axes (blue), minor axes (green) of the braid units, and the braid region guide curve (yellow). (b) A three-strand braid structure representation using helix curves. (c) Grouping the braid units to form the braid strand guide curves. (d) Braid strand guide curve generation result.

the radius of the tube is determined by the minor axis length of the braid units. The generated braid structures are shown in Figure 3.31.

(a) (b) (c)

Figure 3.31: Braid structures. (a) 2D braid strand guide curves. (b) and (c) are the 3D "flat" braid strand guide curves at different views.

3.3.4 Experimental Results

As shown in Figure 3.32, our system can generate a variety of braids given a single-view braided hair image. The amount of braid can be one or more since our braid segmentation can deal with multiple braids. Our results are faithful to the shape of the braids and strands of the input image because of our braid structure analysis. Although the image in our system are all of the back-view (since most of the braids are located at the back of the head), there are no constraints on the views.

To give the braid a more complete look, we have designed a method to generate the root of the braid, as well as the tail of the braid. We first extend the braid region guide curve along its direction towards the braid root, the length of the extension is set as 1/4 of the long axis of the nearest braid unit. Then the endpoint of the extension is set as the root of the braid. However, if this endpoint is outside of the hair region, then we set the intersection point of the braid region guide curve and braid region contour as the root of the braid. We generate the tail of the braid in the same manner. Then we "merge" the braid root pinot by the depth of $-\lambda * b$, and "pop" the braid tail point by the depth of $\lambda * b$. Finally, we connect them with the braid strand guide curves. Since the braid region guide curve has the overall orientation information of the whole braid, and the extension length we used is directly from the image, it gives the braid a more realistic look.

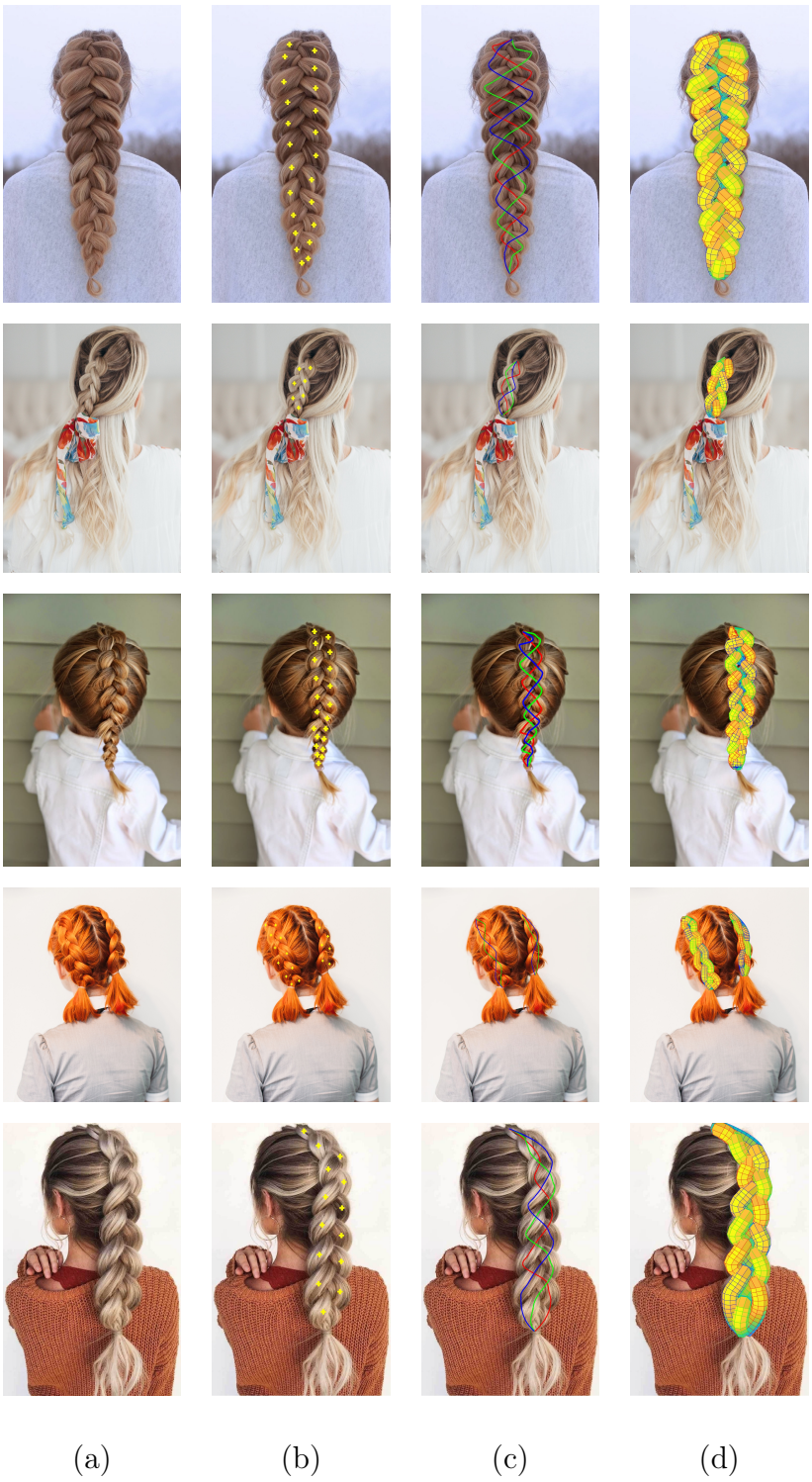


Figure 3.32: 3D braided hair modeling results. (a) Input images. (b) Braid units (The center points of the braid units are shown in yellow dots), (c) Braid strand guide curves with the braid root points and braid tail points. (d) Braided hairstyle structures (without ground-truth 3D hair volume).

3.4 Summary

In this chapter, we elaborate on our works regarding 2D hair analysis and discuss the proposed hair strands extraction system, deep learning models for hairstyle pattern recognition and braid unit recognition, and utilized datasets.

Our 2D hair strand extraction system can automatically extract long hair strands directly from hair images. It plays an important role when constructing 3D hair strands in the non-braid regions of braided hairs.

Our 2D hairstyle pattern recognition system has two parts: image patch-based hairstyle pattern recognition and full hair image hairstyle pattern recognition. The image patch-based hairstyle recognition learns the detailed hairstyle features using deep learning models and recognizes four different hairstyles: straight, curly, kinky, and braided hairs. When given a full hair image, we first divide the hair region into superpixels. Then each superpixel is recognized by our deep learning model. And finally, the recognition results provide the dominant hairstyle as well as the partition of different hairstyles within the hair region. The advantage of using superpixels instead of regular square grids is that the superpixels preserve the nature boundary information of different hairstyle regions, especially for braids. This is the first work that performs hairstyle recognition on both kinky hairs and braided hairs.

We also provide the parametric representation of straight, curly, and kinky hairs as well as the corresponding hairstyle-related parameters. The parameters are used to change the shapes of the hair strands to mimic the reference hair image.

Our deep-learning based braid structure generation system can detect and segment braid units and learn the braid structure by combining the braid unit recognition results

and braiding making rules. As far as we know, this is the first work that can successfully obtain braid structures from a single-view hair image.

Chapter 4

3D Hair Modeling

3D hair capture plays an important role in many computer graphic applications. For example, it can help to provide certain identification and personality to virtual characters in computer games. Another example is in the cosmetic industry, applications that allow the customers to choose suitable hairstyles by applying a variety of hair models on the avatars. However, hair capture remains one of the most challenging tasks due to the characteristics of hair, such as omnipresent occlusion, specula appearance, and complex

discontinuities. In this chapter, we will present our single-view machine-learning-based 3D hair strand modeling system. Furthermore, we introduce our 3D braided hair modeling system and kinky hair modeling system.

4.1 Single-view 3D Hair Strands Reconstruction based on Deep Learning

3D hair strands reconstruction is different from the 3D reconstruction of other human body parts. Unlike other 3D reconstruction methods that focus on recovering the 3D shape of objects, 3D hair strand reconstruction needs to recover individual strands. And this makes 3D hair reconstruction more challenging. The recent success of deep learning brings great improvement in the fields of 3D hair reconstruction. Existing 3D hair reconstruction methods apply deep convolutional neural networks to segment hair regions [12, 51, 57, 3] , estimate hair distribution [12], learn 3D hair model representation [69], generate 3D hair volume [99], and generate 3D hair strands [102]. In this section, we will show our deep-learning based single-view 3D hair strand generation system.

4.1.1 3D Hair Strands Datasets

As we all know, deep learning methods are heavily dependent on data. To train good models, sufficient data are required. However, in the area of hair modeling, building a 3D hair model dataset from scratch is very challenging due to the following reasons:

- plenty of 2D hair images can be downloaded from the Internet, but we don't have access to their 3D volumes or 3D strands.

- existing 3D hair model datasets provide 3D hair models but do not come with real 2D hair images.

- existing 3D hair model datasets lack some complex and constrained hairstyle models, such as the braid hairstyle models, kinky hairstyle models, etc.

As described in Chapter 2, there are limited open-source 3D hair model datasets. In our work, we used the USC-Hair Salon Dataset [36, 80]. The USC-Hair Salon database contains 343 hair models, as shown in Figure 4.1.

Figure 4.1: USC-Hair salon dataset [36, 80]

Table 4.1: Hair model classes. The first row is the hair classes, the second row shows the number of hairs in each class. XS refer to extra short length, S refer to short length. M refer to median length, L refer to long length, and XL refer to extra-long length. The -s, -c refer to straight, curly.

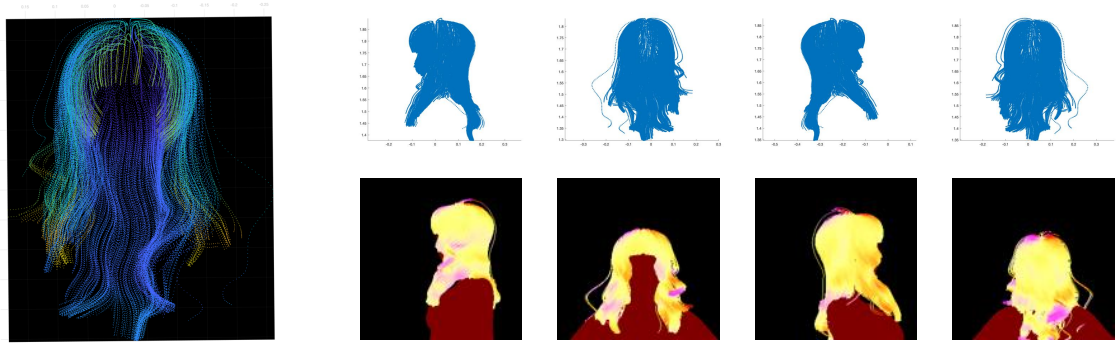
XS-s	S-s	S-c	M-s	M-c	L-s	L-c	XL-s	XL-c
20	110	19	28	65	28	27	23	23

However, 343 3D hair models are not enough for training our 3D hair modeling system. In order to generate more 3D hair models, we adopt a similar method in [102]. We first separate the hairs into 10 classes based on the length and the style of hairs, as shown in Table 4.1. We removed some hair models since their style is ambiguous. Then we cluster the strands of each hair into five guide strands and randomly select the combination of each pair of hair within the same class to generate combinations of new guide hair strands. The generated new guide hair strands are used to guide the generation of more children hair strands. Finally, we generate about 30K 3D hair models. Then we rotate and translate the 3D hairs inside the viewport for a fixed camera and render 4 orientation images at different views. The rotation angles are $[-90°, 0°, 90°, 180°]$ for the yaw axis, and $-15°$ to $15°$ for the pitch and roll axis. Then we create the corresponding orientation images from each view.

Each 3D hair model can be represented as a matrix $H_i(i = 1, .., m)$ with the dimension of $32 \times 32 \times 300$, where 32×32 represents the grid on the scalp as described in [84], and the hair roots locate at the centers of those grids. 300 is the coordinates of a single strands in the order of $[x_1, y_1, z_1, ..., x_{100}, y_{100}, z_{100}]$. Each hair strand is evenly sampled from the root to the end. The generated corresponding orientation image has the dimension of $256 \times 256 \times 3$. An example of the 3D hair is shown in Figure 4.2-(a). The translated and rotated models are shown in the first row of Figure 4.2-(b), and the corresponding orientation images are shown in the second row of Figure 4.2-(b).

4.1.2 3D Guide Hair Strand Generation System

As described in Section 4.1.1, the dataset of our system is a synthetic 3D hair model dataset with corresponding 2D orientation maps rendered from the 3D models, thus when



(a) (b)

Figure 4.2: 2D hair orientation images generated from a 3D hair model at different views. (a) The 3D hair strand model. (b) Hair orientation images that generated from (a) at different view.

training the network, the input image is the hair orientation map. However, when testing, the input is a real-life hair image. In order to overcome this obstacle, we first calculate the orientation map of the input image using the method described in Section 3.1.2, then we generate the human mask and hair mask. By applying those two masks on the orientation map, we can obtain the input image that is suitable for our model.

We designed an encoder-decoder network to generate 3D hair strands from a single-view 2D hair orientation image. The detailed architecture of our guide hair strand generation network is shown in Table 4.2.

In the encoder, we apply convolutional layers to extract high-level features from the input image, then we apply a 2D max-pooling (4x4) to create the latent vector z, then we apply the deconvolutional layers to decode z into the final 3D hair strands. The loss function we used in our network is L_2 reconstruction loss of the 3D position of each strand point, the final loss function is shown as follows:

$$L = \frac{1}{SP}\sum_{i=0}^{S-1}\sum_{j=0}^{P-1}\|C\_predict_{i,j} - C\_gt_{i,j}\|_2^2 \tag{4.1}$$

where S and P are the numbers of strands and the number of points on each strand,

Table 4.2: The architecture of our 3D guide hair strand generation network.

The input size is $256 \times 256 \times 3$

Net	Type	Kernal	Stride	Activation	Output
encoder	conv.	3	2	relu	128, 128, 32
encoder	conv.	3	2	relu	64, 64, 64
encoder	conv.	3	2	relu	32, 32, 128
encoder	conv.	3	2	relu	16, 16, 256
encoder	conv.	3	2	relu	8, 8, 512
encoder	conv.	3	2	relu	4, 4, 1024
encoder	maxpooling	3	2	-	1, 1, 1024
encoder	dense	3	2	relu	1, 1, 4096
encoder	reshape	3	2	-	4, 4, 256
decoder	transconv.	3	2	relu	8, 8, 512
decoder	transconv.	3	2	relu	16, 16, 512
decoder	transconv.	3	2	relu	32, 32, 512
decoder	conv.	1	1	relu	32, 32, 512
decoder	conv.	1	1	tanh	32, 32, 512
decoder	conv.	1	1	-	32, 32, 300

correspondingly. $C\_predict_{i,j}$ is the predicted strand point coordinates and $C\_gt_{i,j}$ is the ground-truth point coordinates. We have trained our network for 450 epochs, and the batch size is 32. We set the learning rate as 0.001. We use Adam [45] for optimization.

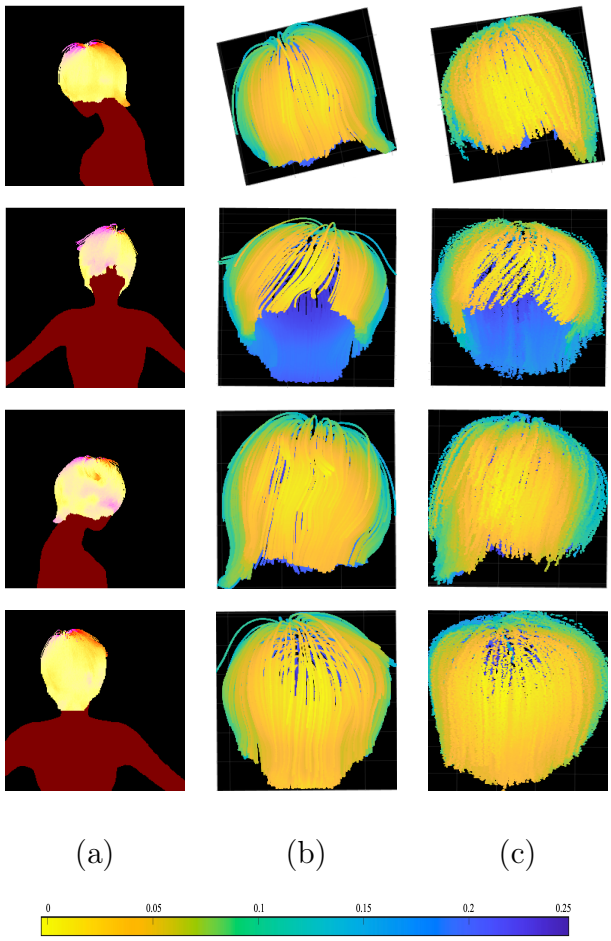


Figure 4.3: 3D hair models generation results. The input images are from the same model of different views. (a) The input images. (b) The generated 3D hair models. (c) The ground-truth 3D hair models. The scale is for

4.1.2.1 Experimental Results

We test our system on a dataset that contains 100 hair models and each hair model has 4 images rendered as described in Section 4.1.1, we have calculated reconstruction error using the position error of all the points on hair strands. One example of generated 3D hair models is shown in Figure 4.3.

We have compared the position errors from the front-view, the side-views, and the back-view of the same hair model. The results indicate that side-view images preserve

more information about both the front-view and back-view, thus the reconstructed models from the side-views have smaller position errors. On the contrary, 3D hair models generated from front-view and back-view both suffer from occlusions, thus they all have larger position errors. The position error results are shown in Table 4.3.

Table 4.3: 3D hair model reconstruction error.

	Front-view	Left-view	Right-view	Back-view
Position Error	0.0235	0.0178	0.0186	0.0276

Our generated 3D hair models with rendering are shown in Figure 4.4. Each hair model contains 1000 hair strands. We use the hair color from the image to render the hair strands. The results show appropriate shapes and lengths of the hair when compared with the input image.

In addition, we want our generated 3D hair model to represent the overall shape of the ground truth. Thus we have conducted experiments for comparing the silhouettes between the predicted hairs and ground-truth hairs. We used the same test set as above. We project the visible part of 3D hair into the $X-Y$ plane, then we extract the contours from both images (the size of the contour image is about 250). We use Chamfer distance to calculate the dissimilarity between two silhouettes. We divided the hair into 4 groups based on the views, the results are shown in Table 4.4.

Table 4.4: Hair silhouette comparison at different views. CD: Chamfer Distance

	Front-view	Left-view	Right-view	Back-view
CD	187.7	267.3	287.5	237.4

Although the generated hairs show appropriate geometry, we found that the out-

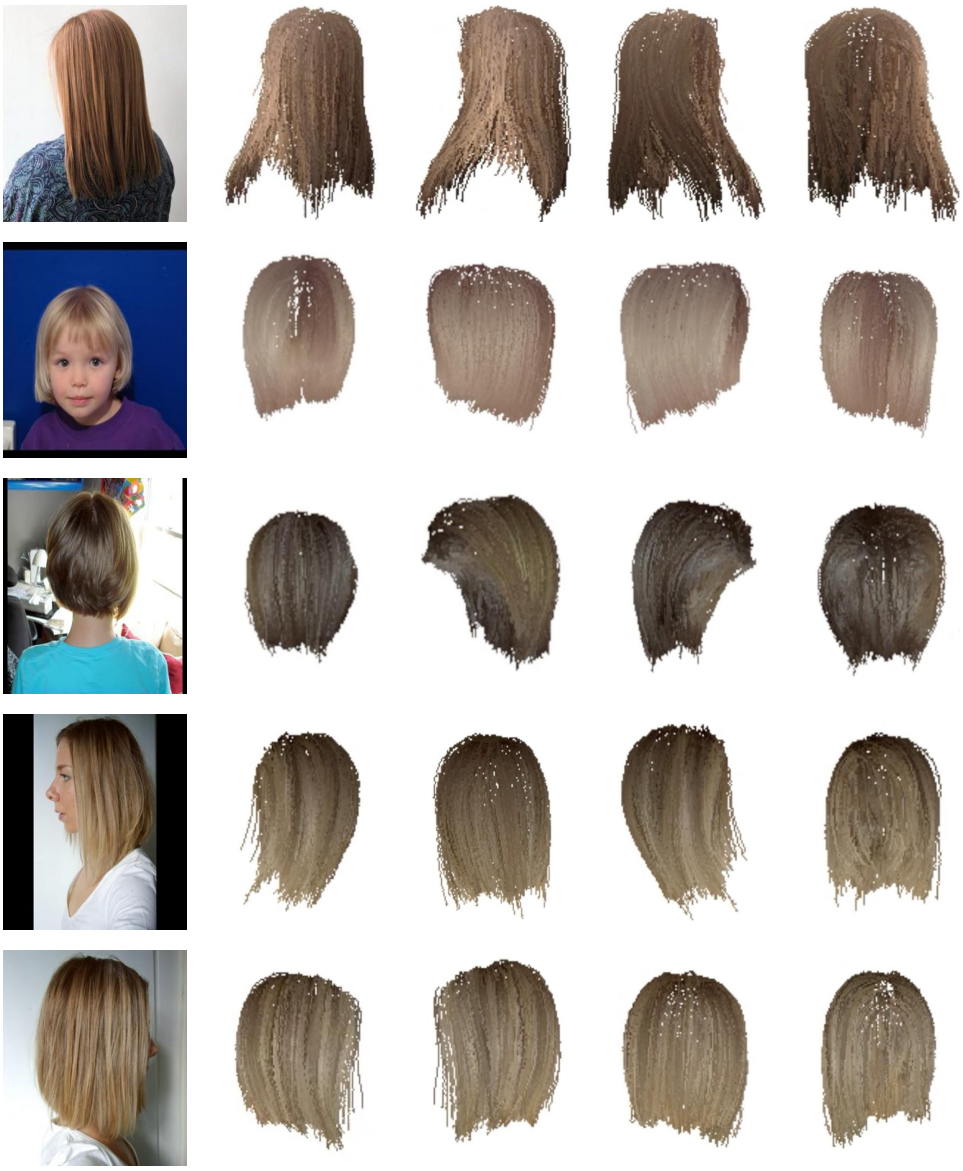


Figure 4.4: 3D hair strands generation results. (a) Single-view input hair images. (b)-(e) Generated hair strands from different views.

put strands have noises and sometimes lose high-frequency details when modeling curly hairstyles. Thus, we adapt our 2D hairstyle analysis method to create curly hairs. We first conduct hairstyle pattern recognition to find the dominant hairstyle of the input image. Then we represent each 3D hair strand using a spline with predefined control

points, this will give the strands a smooth look. Based on the hairstyle recognition results and hairstyle analysis from Section 3.2.4, we estimated the potential set of parameters to control the shape of the hair strands. The experimental results are shown in Figure 4.5.

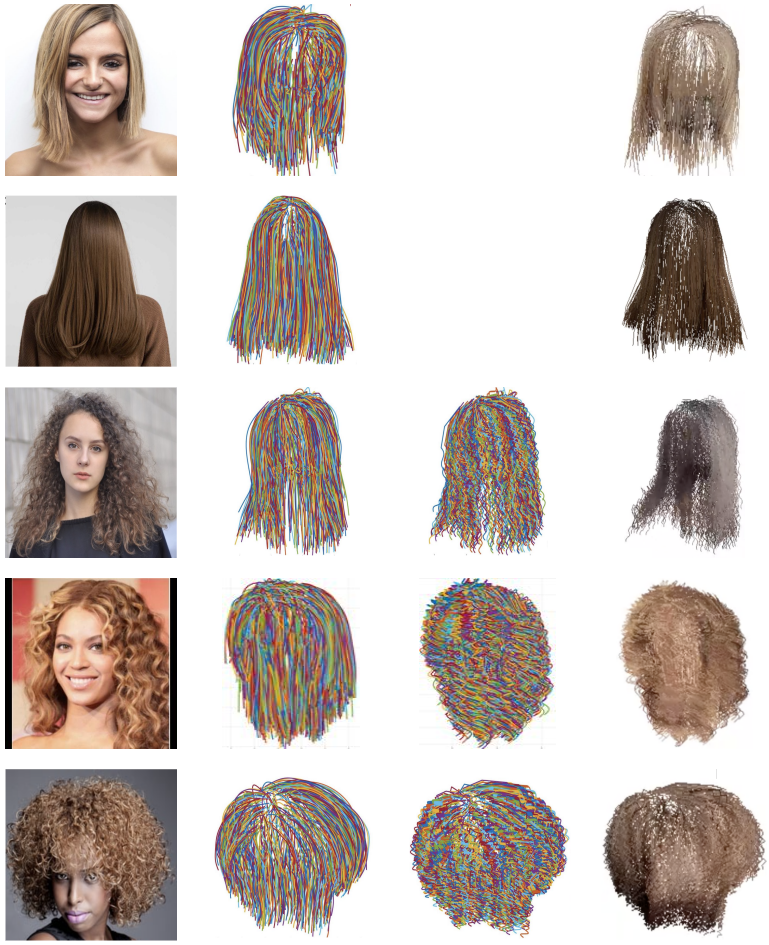


Figure 4.5: Parametric 3D hair models generation. (a) Reference images. (b) Generated 3D model structures. Each hair strand is represented by a spline. (c) Hair strand shapes are controlled by parameters. We didn't apply parameters to changed the styles of two straight hairs, thus the first two rows are blank. (d) Final rendering results.

The parameters we used are: each hair strand is made of 5 segments. Each segment

has only 1 curve, and the amplitude of the curve is 5 pixels. In this thesis work, we focus on parameter-controlled simple hairstyle reconstruction. However, our method can be easily extended to complex hairstyles. Since our hairstyle recognition system not only recognizes the hairstyles but also provides the partitions of each hairstyle, by aligning the generated 3D hair model and hairstyle partition image, we can control the shape of each segment along the hair strand based on the specific hairstyle information.

We have compared our generated 3D hair model with previous works, as shown in Figure 4.6. We can see that our 3D hair models are visually plausible as in the previous works.

4.1.3 Single-view 3D Kinky Hair Modeling

Kinky hairstyle modeling is the most challenging task in the area of hair modeling. Existing hair modeling systems don't include kinky hairs in their works or fail to reconstruct kinky hairstyle [102, 69]. The reasons why kinky hair is so difficult to reconstruct are as follows:

-The characteristics of kinky hair is difficult to capture. The coils of the kinky hair are so tiny that sometimes even the human eye can not distinguish each curl.

-The hair orientation-based methods fail to trace the hair strands based on the 2D hair orientation maps.

-The kinky hair models are not included in existing 3D hair datasets, thus it is very difficult for the data-driven methods/deep learning methods to directly reconstruct kinky hair models.

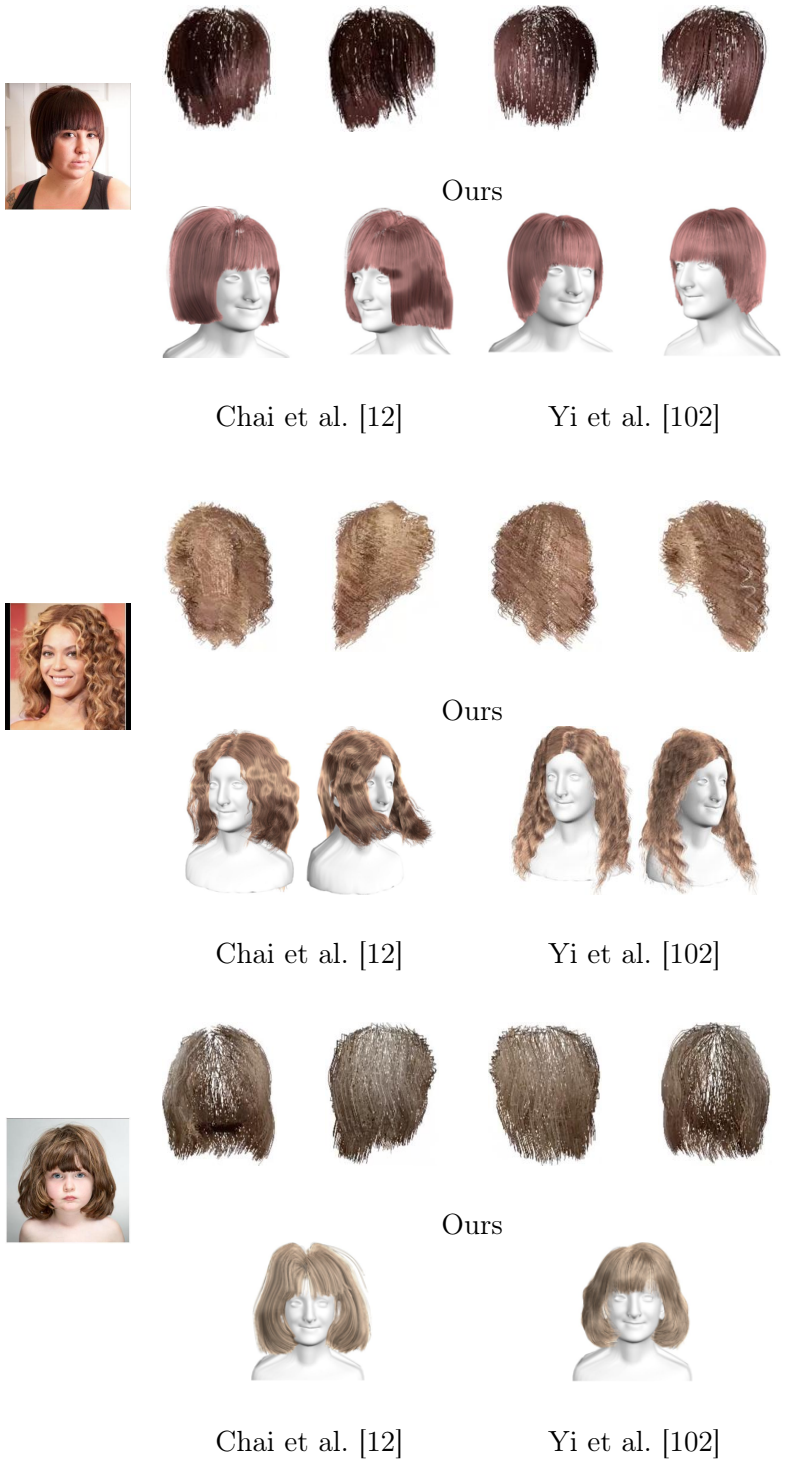


Figure 4.6: Comparison with previous works [12, 102]. Please note that each of our hair model only contain about 1000 3D guide hair strands. The other models contain 9K strands.

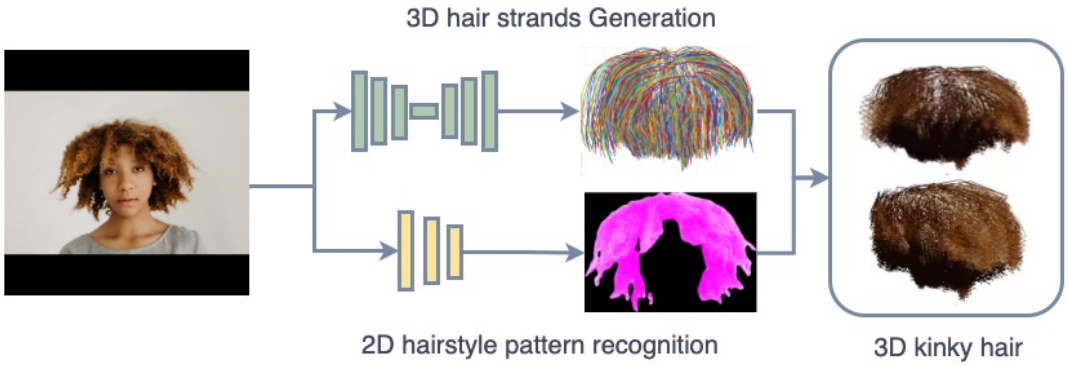


Figure 4.7: Overview of our kinky hair modeling method.

To overcome these challenges, we present our 3D kinky hair modeling system, which combines the 2D hairstyle pattern recognition and the 3D hair strands generation system. The overview of our kinky hair modeling method is shown in Figure 4.7. Given a hair image, we first generate the 3D hair strands using the deep learning model of Section 4.1.2. Then we perform full-image hairstyle recognition to get the hairstyle information. We use the same way to choose the parameters described in Section 3.2.4 and change the structure of the 3D hair strands to make them look like kinky hair. However, since the kinky hair models are extremely difficult to reconstruct, we only aim to obtain visually plausible 3D models.

4.1.3.1 Experimental Results

Our kinky hairstyle modeling results are shown in Figure 4.8. We have chosen kinky hairstyles with different colors, lengths, and coils. The results show that our system can successfully model visual-plausible kinky hairstyles. For the first image, we have 20 segments on each hair strand, the amplitude is 2 and the number of the curve of each segment is 1. For the second image, we have 10 segments on each hair strand, the

amplitude is 2 and the number of the curve of each segment is 1. For the third image, we have 10 segments on each hair strand, the amplitude is 2 and the number of the curve of each segment is 1.

(a) (b) (c) (d) (e)

Figure 4.8: Kinky hair modeling results. (a) Input kinky hairstyle images. (b) - (e) 3D kinky hair models in different views.

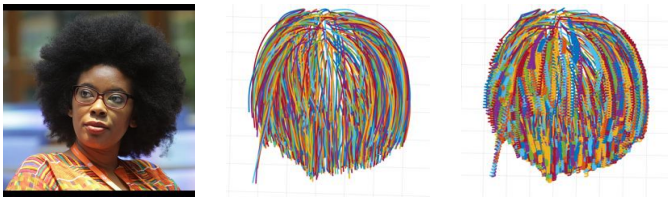
We have also compared our kinky hair reconstruction result with the only result (as failure case) from [69], as shown in Figure 4.9. In Figure 4.9-(b), we can find that even though the rough 3D volume is very similar to the ground truth, however, the generated hair strands are straight hair strands. In Figure 4.9-(c), we have generated the kinky model as the result of the combination of our 2D hair analysis and 3D hair strand generation. Our kinky hair model is more visually plausible.

Our work has limitations too. Here is a failure case of the kinky hairstyles as shown in Figure 4.10. This is a common limitation of the data-driven method. Because there is no hairstyle model with hair strands that go upward in the 3D hairstyle dataset for

(a) (b) (c)

Figure 4.9: Kinky hair modeling comparison. (a) The input image. (b) The kinky hair model of [69] (c) Our reconstructed kinky hair.

training.



(a) (b) (c)

Figure 4.10: A failure case of kinky hair reconstruction. (a) The input image. (b) The generated 3D guide strands. (c) The 3D kinky hair model structure.

4.2 Single-view 3D Braided Hair Modeling

3D braided hair modeling is a crucial and challenging problem that hasn't been explored much. It's difficult to reconstruct braids for the following reasons:

-There is no 3D braided hair dataset for training deep learning models.

-The braid structure is very difficult to obtain directly from the image since it is highly complicated.

-The existence of braids changes the appearance of the surrounding strands and makes it even more difficult for modeling.

In Section 3.3, we have obtained the braid structure from a 2D hair image, now we need to 'pop' the braid on the 3D hair strands. We have developed several strategies separately to deal with different braided hairs. We first describe two main scenarios as follows:

- The braid has minimum impact on the shape of the rest of the hair strands. The braid "floats" above the hair strands.

- The braid dramatically changes the shape of other hair strands. Furthermore, hair strands can also be separated into several sections due to multiple braids, etc.

4.2.1 Braided Hair Modeling based on 3D Guide Strand Generation

For the first scenario, we combined our deep-learning based hair models of Section 4.1 with our generation 2D hair braid structure of Section 3.3 to generate the final braided hair models. The overview is shown in Figure 4.11.

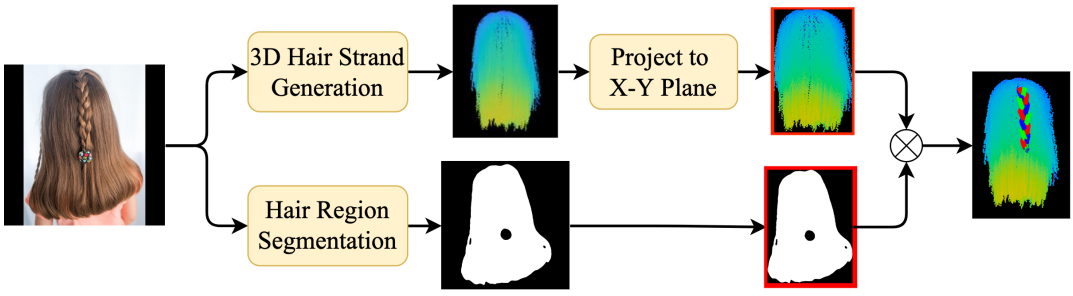


Figure 4.11: The overview of the single-view braided hair modeling in first scenario.

Given the input braid image, we first generate the 3D hair strands, then project them into the $X - Y$ plane by orthographic projection to create a hair mask. We generate the bounding box around this hair mask as $bbox_1$. We also have the hair region bounding box based on hair segmentation as $bbox_2$. We align those two bounding boxes together to estimate the location of the braid in the $bbox_1$. For each 2D braid point p, we find 3 nearest projected points inside $bbox_1$and calculate the z coordinate of p by linear interpolation of z coordinates of those 3D hair strands points. The experiment results are shown in Figure 4.12.

(a) (b) (c) (d)

Figure 4.12: 3D braided hair modeling result. (a) Input braided hairstyle image. (b) 2D braid structure. (c) 3D hair strands. (4) 3D braided hair modeling result. The braid strands are represented using 3 tubes in different colors (red, green, and blue).

However, the second scenario is more challenging. Our 3D hair strand generation system fails to generate the correct 3D hair models, as shown in Figure 4.13.

In Figure 4.13-(b), the 3D hair strands are separated into two groups to represent the two braids, it is not suitable for generating visually plausible 3D braided hairs. Thus we present a silhouette-based 3D braided hair modeling system to overcome those challenges.

(a) (b)

Figure 4.13: A failure case when generating 3D braided hair. (a) Input braided hairstyle image with 2 braids. (b) Generated 3D hairs.

4.2.2 Silhouette-based 3D Braided Hair Modeling

When the braids dramatically change the shape of hair strands in the non-braid region, we have developed a silhouette-based 3D braided hair modeling system to deal with this scenario. The overview of our silhouette-based 3D braided hair modeling system is shown in Figure 4.14. As shown in Figure 4.14, we first generate the braid based on our 2D braid structure analysis (Section 3.3) and 2D hair strands of non-braid region using our 2D hair extraction methods (Section 3.1). Then we uplift the 2D hair strands and 2D braids to 3D to obtain the complete braided hair model. Recovering a 3D volume from a single image is an ill-posed problem. It is especially difficult in our case since the braid is usually at the back of the head. To overcome this difficulty, we design a new approach utilizing the silhouette of the hair region to create the 3D hair volume.

By carefully examining the relative locations of braids with the head, we can divide the braids into two categories. And for each category, we use different methods for silhouette selection.

- Part of the braid is attached to the head (from the start of the braid till the nape

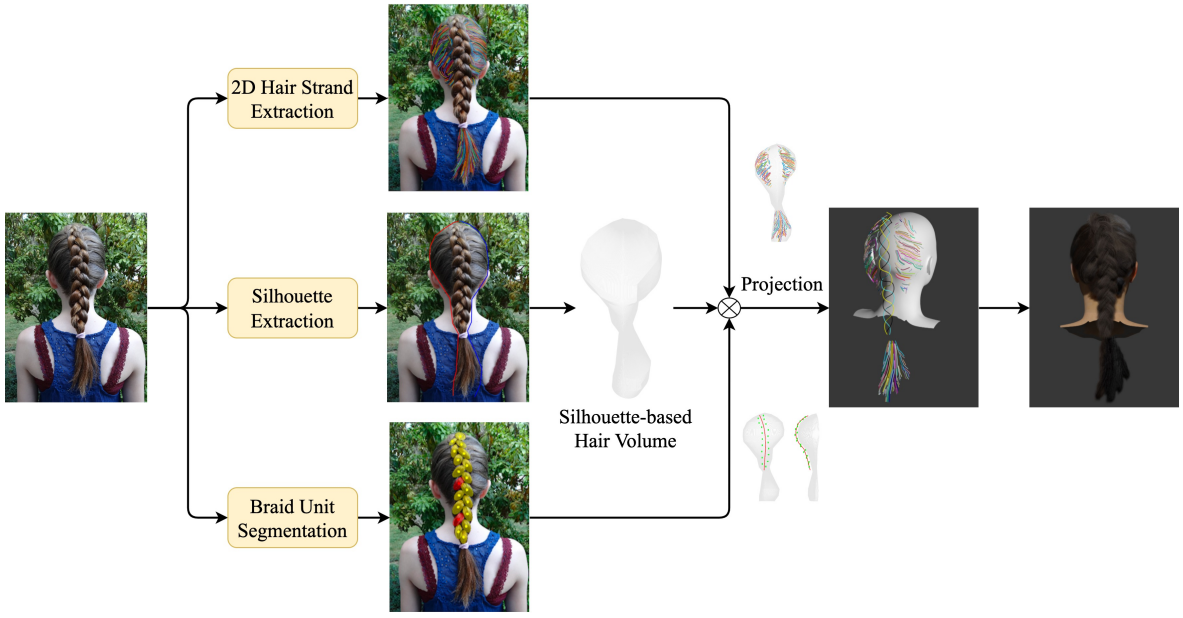


Figure 4.14: The overview of silhouette-based 3D braided hair modeling system.

of the neck) since small pieces of hair from both sides of the head are brought into the braid. We manually select 10 points $\mathrm{lp}_i, (i = 1, \ldots, 10)$ along the left side of the hair silhouette and 10 points $\mathrm{rp}_j, (j = 1, \ldots, 10)$ along the right side of the hair silhouette.

- Only one end of the braid is attached to the head (usually the top end, and the braid hangs down due to gravity), we manually select 5 points along the left side of the hair silhouette from the top point until the left-most point, then select other 5 points that share the same x coordinate as the 5th point. Those 10 points $\mathrm{lp}_i, (i = 1, \ldots, 10)$ represent the left silhouette. We select 10 points $\mathrm{rp}_j, (j = 1, \ldots, 10)$ on the right side in the same manner for the right silhouette. The final shape of the silhouette is similar to an arch.

Then we interpolate 40 points between each pair of adjacent points among lp_i to represent the left silhouette $\mathrm{S}_{\mathrm{left}}$, and 40 points between each pair of adjacent points among rp_j to

represent the right silhouette S_{right}, correspondingly. Furthermore, based on the first 3 points from both the left and right silhouette, we calculate 3 middle points between the corresponding lp_k and $rp_k, (k = 1, 2, 3)$. Then we fit a straight line based on those three points as the center line of the hair region.

We set the image on the X-Y plane, and select the rotation axis based on the center line. Then the left silhouette S_{left} and the right silhouette S_{right} are rotated around the rotation axis π radians toward each other by the interval of intvl (intvl = 0.01 radians in our experiments) to generate two groups of 3D curves: Sl_θ and Sr_θ, where $(\theta = 0, 0.01, \ldots, \pi)$ is the rotation angle. Then we perform linear interpolation between the corresponding left curves and right curves to generate the final 3D curves S_θ following Equation 4.2. These resulting volume and uplifted data are shown in Figure 4.15-(c).

$$S_\theta = \frac{\pi - \theta}{\pi} \times Sl_\theta + \frac{\theta}{\pi} \times Sr_\theta \tag{4.2}$$

Then we project the 3D hair volume back to the X-Y plane by orthographic projection. For each 2D hair data point p in the 2D image, we find 3 nearest projected points and calculate the z coordinate of p by linear interpolation of the z coordinates of those 3D points in the 3D hair volume.

Braids stand on top of the other hair strands. We thus pop-up the 3D braid region center curve and 3D braid unit center points from our hair volume. The distance between the braid and the hair volume is calculated by averaging the minor axis length of all braid units. Results are shown in Figure 4.15-(c)-(e).

4.2.2.1 Dense Hair Strands Generation and Rendering

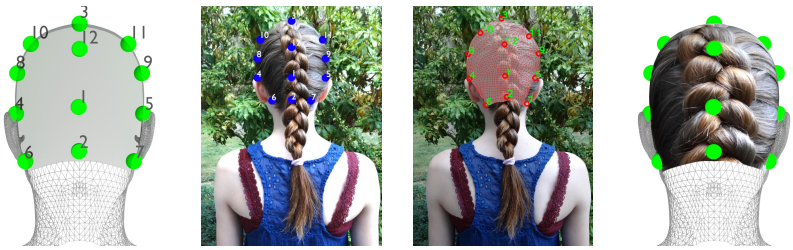
The 3D curves act as guide curves, and we use a particle system that generates hair strands based on a user controlled density. Triangles from the scalp of an existing digital

(a) (b) (c) (d) (e)

Figure 4.15: 3D volume, 3D hair strands and 3D braids information. (a) The left and right silhouettes based on the hair region. (b) Surface of revolution created by rotating the silhouette curves (as well as the linear interpolation of the curves). (c) Projected 3D hair strands. (d) Projected 3D braid unit center points and braid region center curve. (e) Helical curves (in red) follow the braid center curve.

avatar are used as emitters for dense hair strands. The guide curves are typically different from the scale of the digital avatar. Consequently, the first step is to manually scale, rotate, and translate the guide curves to fit the scalp. The closest triangle in the scalp mesh is found for each of the guide curves. For each guide curve, the hair strands are emitted from this triangle and they follow the guide curve. In a similar way, we generate dense hair strands for the braids based on the guide helical curves.

The multiple strands are assigned the colors from the input image, by using this image as a texture for the scalp mesh. We use a standard 2D parametrization to get a flattened version of the hair root region of the scalp mesh. We then deform the scalp mesh (using Laplacian surface editing) so that the contour of the mesh fits the hair region in the image. We do so by manually setting sparse corresponding landmarks (about 12) on the contour of the 3D hair root mesh as well as on the image (see Figure 4.16-(a) and (b)). Each of the dense hair strands then gets its color from the texture color at the UV coordinate where the strand is emitted. [1]



(a) (b) (c) (d)

Figure 4.16: Texture mapping of hair region. The scalp (a) and the image (b) are marked with landmarks. Using Laplacian surface editing, the contour of the hair-root mesh is aligned (c) with the hair region of the image. (d) shows the hair region of the image applied as the texture of the scalp.

4.2.2.2 Experimental Results

The reconstructed 3D hair strands and 3D braids structures are shown in Figure 4.17. And more rendered 3D braided hair models are shown in Figure 4.18.

Our results showcase how the procedural modeling is faithful to the shape of the hairstyle and braids. The input braid hair images are different in hair strand length, hair color, braid styles, and the number of braids. Our approach is successful for all these variations. It is interesting to note that our approach can detect multiple braids (see 6th photograph from Figure 4.18) from a single image. [37] is the only other work also capturing braids. Their results are slightly better than ours, but this comes at the expense of a much more complex capture setup: capturing multiple RGB-D images by slowly moving a hand-held Kinect camera around the hair. With a single RGB image our approach can reconstruct most of the details found in their results.

We have also compared the two 3D shapes of braided hairs (as the first scenario men-

[1]This section is collaboration work with Srinivasan Ramachandran, Ph.D. student at École de technologie supérieure

Figure 4.17: 3D braids and 3D hair strands structures. The first row shows the reference images. The second row shows the braid structure in 2D. The last row shows the 3D structures of the braids and hair strands. The braid (yellow) and hair strands (blue) are represented using different colors. Lighter color indicates the object is near to the viewer, darker color means the object is far away from the viewer.

tioned above) from our deep learning based methods and the silhouette-based methods, as shown in Figure 4.19. The hair shape in Figure 4.19-(b) is more visually plausible since it is generated strictly based on the image information. However, the hair model in Figure 4.19-(c) is a complete 3d hair model.

Limitation Our approach has some limitations. Currently, our braided hair reconstruction generates 3-strand braids. This could be improved by adding a braid styles recognition method to perform n-strand braid reconstruction. The presence of accessories like hair clips and bands leads to discontinuities in the braid region as seen in Figure 4.20. Typically the discontinuity happens towards the end of the braids and

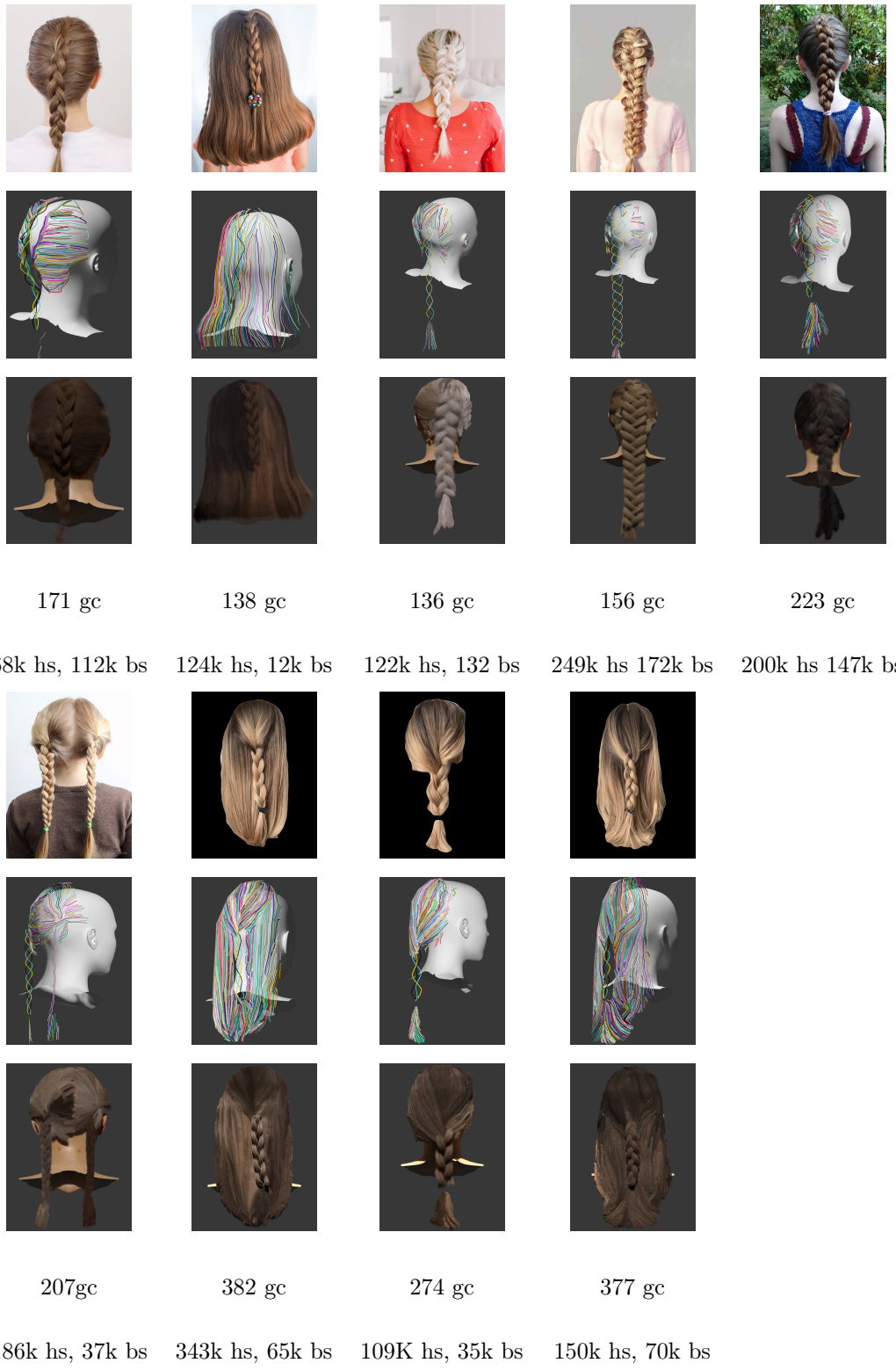
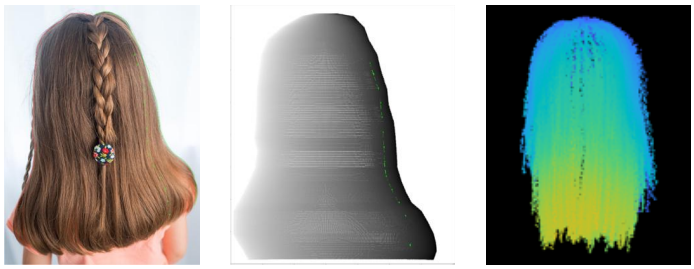


Figure 4.18: Rendered 3D braided hair models. The 1st and 4th rows shows the input images. The 2nd and 5th rows contains the resulting guide hair curves. The 3rd and 6th rows show the procedural modeling and rendering. **gc**: guide curves, **hs**: hair strands, **bs**: braid strands.



(a) (b) (c)

Figure 4.19: 3D hair shape comparison. (a) The reference image. (b) The silhouette based 3D hair volume. (c) The 3D strands generated by our machine learning model(4.1.2).

separates the fishtail region.

Figure 4.20: Example of hair accessory leading to a discontinuity in the braid region.

Another failure case is shown in Figure 4.21. Although we can reconstruct the 3D braid, we notice that some hair strands are not correctly extracted from the hair image, resulting in floating hair segments on the face area. This is because the hair region segmentation method fails to segment the hair strands on the face as shown in the red box. Hair color is not perfectly consistent with the image. The color of each strand is constant. As such, the only color variation along a strand is caused by the lighting. Furthermore, our hair color extraction approach does not support hair relighting very well. As we do not reconstruct proper depth from the image, the 3D shape of the hair is not always faithful to what would be expected. Finally, we do not reconstruct a full

hair, so we are missing information for the parts of the head not shown in the image.

(a) (b) (c)

Figure 4.21: Hair strand reconstruction failure case. (a) the reference hair image. (b) the rendered 3D hair model. (c) hair region segmentation result.

4.3 Summary

In this chapter, we focus on generating the 3D hair models from single-view hair images. We designed an encoder-decoder deep learning model to reconstruct the 3D hair strand structures for straight, curly, and kinky hairs. Then we apply the hairstyle-related parameters that were learned directly from the hair image to change the shapes and curliness of 3D hair strands to make the generated hair model resemble the input hair image. Our work is the first work that can reconstruct visual-plausible 3D kinky hair from a single-view image. The success of creating the kinky hairstyle can not be achieved without our 2D hairstyle pattern analysis. For braided hairs, if the braids don't change the shape or orientation of the non-braid region, then we combine our 2D braid structure generation (Section 3.3) and 3D hair strands generation to create the complete 3D braided hair models. However, if the braids change the shape of the non-braid region dramatically, we first recover the 3D hair volume using the silhouette of the hair region. Then we generate the hair strands in the non-braid region using (Section 3.1) and 2D

braid structure. And finally, we project the 2D hair strands and 2D braid onto the 3D hair volume to obtain the final 3D braided models. Our work is the first work that can generate complete braided hair models based on a single-view hair image.

Chapter 5

Conclusion

In this thesis, we proposed a system to address the task of single-view 3D hair modeling by utilizing both 2D hair analysis and 3D hair strand generation. Our approach is a combination of image processing, deep neural networks, as well as 2D and 3D geometric algorithms. Our experimental results show that our system can generate visually plausible 3D hair models of various hairstyles from single-view hair image.

5.1 Summary of Contributions

For 2D hair analysis, we addressed three tasks: (1) 2D hair stands extraction, (b) 2D hairstyle pattern recognition, and (3) 2D braid structure analysis. We proposed an effective and efficient 2D hair strand extraction algorithm based on orientation maps and our image-enhancement method to obtain long and visual-feasible hair strands from images. We also designed a deep learning network that is trained on our hairstyle pattern dataset to learn the hairstyle patterns and the distributions of four basic hairstyles (straight, curly, kinky, and braid) from hair images. This is the first work that performs hairstyle recognition on both kinky hairs and braided hairs. From this work, we learned the attributes of different hairstyles. In addition, we have proposed a deep learning based braid structure generation system. We trained a model to detect and segment braid units and learn the braid structure by combining the braid unit recognition results and braiding making rules. As far as we know, this is the first work that can successfully obtain braid structures from a single-view hair image.

For 3D hair modeling, unlike existing single hair reconstruction systems, our single-view 3D hair modeling system is a combination of 2D hair analysis and deep learning based 3D hair strand generation models. Our system can deal with different hairstyles and take approaches fitting to the characteristics. In addition, our system has fewer view constrain than existing systems, which means we can reconstruct 3D hair models even when there is no human face in the input image.

Furthermore, we are the first work to reconstruct 3D braided hair models from single-view hair images. For braided hair modeling, we first recognize and segment braid units and extract the braid structure from the hair image. Then we focused on 3D braid

reconstruction using helix curves whose parameters are obtained by the measurements from a 2D image while other hairstyle regions are also reconstructed by extracting strands directly from the image. We have tested with several images and the experiment results show that our method has a high potential for a bright future of hair modeling with easy input and output of, firstly, automatic plausible hair modeling where the cloning of a hair is adapted and, secondly, procedural modeling of hair where hair length or braid frequency and amplitudes easily modify the output hair. Moreover, our work is the first work that can reconstruct visual-plausible 3D kinky hair from single-view image. The success of creating the kinky hairstyle can not be achieved without our 2D hairstyle pattern analysis.

At last but not least, we have also created two datasets: the hairstyle pattern image dataset, and the braid unit dataset. Those two datasets can be used by future hair-related research works.

5.1.1 Future Works

Although our work can generate visual-plausible 3D hair models of various hairstyle from single-view hair images, there are still several directions that we can improve our works.

Hairstyle Pattern Recognition In this work, we divide the hairstyles into four categories: straight, curly, braid, and kinky. This can be improved by including: (1) more hairstyles, such as fulani braids, bantu knots, and dreadlocks, etc, as shown in Figure 5.1. (2) more detailed classifications within each hairstyle. For example, we can further divide the curly hairstyle based on curliness, etc.

Braid Strands Amount In this work, we mainly focus on 3 strands braids extraction. For more complex braids with more braid strands or more complicated braiding

(a) (b) (c)

Figure 5.1: More hairstyles that can be included in the hairstyle recognition. (a) fulani braids, (b) bantu knots, (c) dreadlock.

methods, we may need to combine more information to obtain correct braid structures. One possible way is to design a braid type recognition system that can recognize different kinds of braids, such as French braid, Dutch braid, fishtail braid, etc. Then by combining the braid type information and the braid unit segmentation information, we can obtain the braid structure for a large variety of braids.

Braid Amount When there are multiple braids in the input hair image, we manually separate the braids and extract the braid structures correspondingly. Our work can be improved to automatically separate multiple non-overlapping braids by clustering the braid units based on their locations.

3D Hair Strand Generation Single view reconstruction is an ill-posed problem, especially when we try to reveal the hidden view. As shown in Figure 5.2, the back part of the hair is slightly longer than what we have estimated. This may be caused by some ambiguous styles in the training dataset.

(a) (b) (c) (d) (e)

Figure 5.2: Hidden-view of 3D hair strands. (a) is the input image. (b)-(e) are the generated 3D hairs in different views.

References

[1] Aarabi, Parham. "Automatic Segmentation of Hair in Images". In: *IEEE International Symposium on Multimedia*. 2015, pp. 69–72.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. "SLIC superpixels Compared to State-of-the-art Superpixel Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282.

[3] Tudor Alexandru Ileni, Diana Laura Borza, and Adrian Sergiu Darabant. "Fast In-the-Wild Hair Segmentation and Color Classification". In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. 2019, pp. 59–66.

[4] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1534–1543.

[5] Thabo Beeler, Bernd Bickel, Gioacchino Noris, Paul Beardsley, Steve Marschner, Robert W Sumner, and Markus Gross. "Coupled 3D Reconstruction of Sparse Facial Hair and Skin". In: *ACM Transactions on Graphics* 31.4 (2012), pp. 1–10.

[6] Dimitri P. Bertsekas. "A Distributed Asynchronous Relaxation Algorithm for the Assignment Problem". In: *IEEE Conference on Decision and Control*. 1985, pp. 1703–1704.

[7] Christopher M. Bishop. “Mixture Density Networks”. 1994.

[8] Volker Blanz and Thomas Vetter. “A Morphable Model for the Synthesis of 3D Faces”. In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. 1999, pp. 187–194.

[9] Xudong Cao, Yichen Wei, Fang Wen, and Jian Sun. “Face Alignment by Explicit Shape Regression”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2887–2894.

[10] Yan-Pei Cao, Zheng-Ning Liu, Zheng-Fei Kuang, Leif Kobbelt, and Shi-Min Hu. “Learning to Reconstruct High-quality 3D Shapes with Cascaded Fully Convolutional Networks”. In: *Proceedings of the European Conference on Computer Vision*. 2018, pp. 616–633.

[11] Menglei Chai, Linjie Luo, Kalyan Sunkavalli, Nathan Carr, Sunil Hadap, and Kun Zhou. “High-Quality Hair Modeling from a Single Portrait Photo”. In: *ACM Transactions on Graphics* 34.6 (2015).

[12] Menglei Chai, Tianjia Shao, Hongzhi Wu, Yanlin Weng, and Kun Zhou. “AutoHair: Fully Automatic Hair Modeling from a Single Image”. In: *ACM Transactions on Graphics* 35.4 (2016).

[13] Menglei Chai, Lvdi Wang, Yanlin Weng, Xiaogang Jin, and Kun Zhou. “Dynamic Hair Manipulation in Images and Videos”. In: *ACM Transactions on Graphics* 32.4 (2013), pp. 1–8.

[14] Menglei Chai, Lvdi Wang, Yanlin Weng, Yizhou Yu, Baining Guo, and Kun Zhou. “Single-view Hair Modeling for Portrait Manipulation”. In: *ACM Transactions on Graphics* 31.4 (2012), pp. 1–8.

[15] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *Proceedings of the European Conference on Computer Vision*. 2016, pp. 628–644.

[16] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2017.

[17] Jyotikrishna Dass, Monika Sharma, Ehtesham Hassan, and Hiranmay Ghosh. "A Density based Method for Automatic Hairstyle Discovery and Recognition". In: *The Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*. 2013, pp. 1–4.

[18] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 1486–1494.

[19] Gayane Dolyan Descornet. *Andre Walker Hair Typing System - Women Health Info Blog*. 2017. URL: `https://www.women-info.com/en/andre-walker-hair-typing-system/`.

[20] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. "Learning to Generate Vhairs with Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1538–1546.

[21] Abhishek Dutta and Andrew Zisserman. "The VIA annotation software for images, audio and video". In: *Proceedings of the 27th ACM international conference on multimedia*. 2019, pp. 2276–2279.

[22] Haoqiang Fan, Hao Su, and Leonidas J Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 605–613.

[23] *Frostbite*. `https://www.ea.com/frostbite/news/frostbite-hair-rendering-and-simulation-2`.

[24] Yasutaka Furukawa and Jean Ponce. "Accurate, Dense, and Robust Multiview Stereopsis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.8 (2010), pp. 1362–1376.

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361.

[26] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. "Learning a Predictable and Generative Vector Representation for Objects". In: *Proceedings of the European Conference on Computer Vision*. 2016, pp. 484–499.

[27] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: `1406.2661 [stat.ML]`.

[28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets".

In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014, pp. 2672–2680.

[29] Wenzhangzhi Guo and Parham Aarabi. "Hair Segmentation Using Heuristically-Trained Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.1 (2018), pp. 25–36.

[30] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. "High-Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference". In: *The IEEE International Conference on Computer Vision*. 2017.

[31] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision*. 2017, pp. 2980–2988.

[32] Lin Hong, Yifei Wan, and Anil Jain. "Fingerprint Image Enhancement: Algorithm and Performance Evaluation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), pp. 777–789.

[33] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: abs/1704.04861 (2017).

[34] Liwen Hu, Derek Bradley, Hao Li, and Thabo Beeler. "Simulation-Ready Hair Capture". In: *Comput. Graph. Forum* 36.2 (2017), pp. 281–294.

[35] Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. "Robust Hair Capture Using Simulated Examples". In: *ACM Transactions on Graphics* 33.4 (2014).

[36] Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. "Single-view Hair Modeling using a Hairstyle Database". In: *ACM Transactions on Graphics* 34.4 (2015), pp. 1–9.

[37] Liwen Hu, Chongyang Ma, Linjie Luo, Li-Yi Wei, and Hao Li. "Capturing Braided Hairstyles". In: *ACM Transactions on Graphics* 33.6 (2014).

[38] Liwen Hu, Shunsuke Saito, Lingyu Wei, Koki Nagano, Jaewoo Seo, Jens Fursund, Iman Sadeghi, Carrie Sun, Yen-Chun Chen, and Hao Li. "Avatar Digitization from a Single Image for Real-Time Rendering". In: *ACM Transactions on Graphics.* 36.6 (2017).

[39] Gary B. Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. "Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments". In: *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition.* 2008.

[40] Gary B Huang and Erik Learned-Miller. "Labeled Faces in the Wild: Updates and New Reporting Procedures". In: *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep* 14.003 (2014).

[41] Anil K. Jain and Farshid Farrokhnia. "Unsupervised Texture Segmentation Using Gabor Filters". In: *IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings.* 1990, pp. 14–19.

[42] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. "Gal: Geometric Adversarial Loss for Single-view 3D-object Reconstruction". In: *Proceedings of the European conference on computer vision.* 2018, pp. 802–816.

[43] Antonio Torralba Joseph J. Lim Hamed Pirsiavash. "Parsing IKEA Objects: Fine Pose Estimation". In: *IEEE International Conference on Computer Vision*. 2013, pp. 2992–2999.

[44] Khalil Khan, Massimo Mauro, and Riccardo Leonardi. "Multi-class Semantic Segmentation of Faces". In: *2015 IEEE International Conference on Image Processing*. 2015, pp. 827–831.

[45] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[46] Diederik P Kingma and Max Welling. "Auto-encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[47] Abhijit Kundu, Yin Li, and James M Rehg. "3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3559–3568.

[48] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. "Autoencoding beyond Pixels Using a Learned Similarity Metric". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. 2016, pp. 1558–1566.

[49] Aldo Laurentini. "The Visual Hull Concept for Silhouette-based Image Understanding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.2 (1994), pp. 150–162.

[50] Kuang-chih Lee, Dragomir Anguelov, Baris Sumengen, and Salih Burak Gokturk. "Markov Random Field Models for Hair and Face Segmentation". In: *IEEE International Conference on Automatic Face Gesture Recognition*. 2008, pp. 1–6.

[51] Alex Levinshtein, Cheng Chang, Edmund Phung, Irina Kezele, Wenzhangzhi Guo, and Parham Aarabi. “Real-Time Deep Hair Matting on Mobile Devices”. In: *15th Conference on Computer and Robot Vision* (2017), pp. 1–7.

[52] Shu Liang, Xiufeng Huang, Xianyu Meng, Kunyao Chen, Linda G. Shapiro, and Ira Kemelmacher-Shlizerman. “Video to Fully Automatic 3D Hair Model”. In: *ACM Transactions on Graphics* 37.6 (2018). ISSN: 0730-0301.

[53] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. “Microsoft coco: Common Objects in Context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[54] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision*. 2015.

[55] Linjie Luo, Hao Li, Sylvain Paris, Thibaut Weise, Mark Pauly, and Szymon Rusinkiewicz. “Multi-view Hair Capture using Orientation Fields”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1490–1497.

[56] Linjie Luo, Hao Li, and Szymon Rusinkiewicz. “Structure-Aware Hair Capture”. In: *ACM Transactions on Graphics* 32.4 (2013).

[57] Yuanxi Ma, Cen Wang, Shiying Li, and Jingyi Yu. “Hair Segmentation on Time-of-Flight RGBD Images”. In: *CoRR* abs/1903.02775 (2019).

[58] Priyanka Mandikal and R. Venkatesh Babu. “Dense 3D Point Cloud Reconstruction Using a Deep Pyramid Network”. In: *2019 IEEE Winter Conference on Applications of Computer Vision*. 2019, pp. 1052–1060.

[59] Priyanka Mandikal, K L Navaneet, Mayank Agarwal, and R. Venkatesh Babu. “3D-LMNet: Latent Embedding Matching for Accurate and Diverse 3D Point Cloud Reconstruction from a Single Image”. In: *Proceedings of the British Machine Vision Conference*. 2018.

[60] Umar Riaz Muhammad, Michele Svanera, Riccardo Leonardi, and Sergio Benini. “Hair Detection, Segmentation, and Hairstyle Classification in the Wild”. In: *Image and Vision Computing* 71 (2018), pp. 25–37. ISSN: 0262-8856.

[61] Giljoo Nam, Chenglei Wu, Min H. Kim, and Yaser Sheikh. “Strand-Accurate Multi-View Hair Capture”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[62] Sylvain Paris, Hector M. Briceño, and François X. Sillion. “Capture of Hair Geometry from Multiple Images”. In: *ACM Transactions on Graphics* 23.3 (2004), pp. 712–719. ISSN: 0730-0301.

[63] Sylvain Paris, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. “Hair Photobooth: Geometric and Photometric Acquisition of Real Hairstyles”. In: *ACM Transactions on Graphics* 27.3 (2008), pp. 1–9.

[64] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 77–85.

[65] Siyang Qin, Seongdo Kim, and Roberto Manduchi. "Automatic Skin and Hair Masking using Fully Convolutional Networks". In: *2017 IEEE International Conference on Multimedia and Expo*. 2017, pp. 103–108.

[66] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. "OctNet: Learning Deep 3D Representations at High Resolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6620–6629.

[67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). eprint: 1505.04597.

[68] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252. ISSN: 0920-5691.

[69] Shunsuke Saito, Liwen Hu, Chongyang Ma, Hikaru Ibayashi, Linjie Luo, and Hao Li. "3D Hair Synthesis Using Volumetric Variational Autoencoders". In: *ACM Transactions on Graphics* 37.6 (2018).

[70] Yuefan Shen, Changgeng Zhang, Hongbo Fu, Kun Zhou, and Youyi Zheng. "DeepSketchHair: Deep Sketch-Based 3D Hair Modeling". In: *IEEE Transactions on Visualization and Computer Graphics* 27.7 (2021), pp. 3250–3263.

[71] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor Segmentation and Support Inference from RGBD Images". In: *Proceedings of the European Conference on Computer Vision*. 2012, pp. 746–760.

[72] Edward Smith and David Meger. "Improved Adversarial Systems for 3D Object Generation and Reconstruction". In: *CoRR* abs/1707.09557 (2017). arXiv: 1707.09557.

[73] Chao Sun, Fatemeh Cheraghchi, and Won-Sook Lee. "2D Hair Strands Generation based on Template Matching". In: *International Conference On Simulation and Modeling Methodologies, Technologies and Applications*. 2014, pp. 261–266.

[74] Chao Sun and Won-Sook Lee. "Hairstyle Pattern Recognition based on CNNs". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics*. 2017, pp. 1840–1845.

[75] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. "Pix3d: Dataset and Methods for Single-image 3D Shape Modeling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2974–2983.

[76] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[77] Richard Szeliski and P. H. S. Torr. "Geometrically Constrained Structure from Motion: Points on Planes". In: *IN EUROPEAN WORKSHOP ON 3D STRUCTURE*

FROM MULTIPLE IMAGES OF LARGE-SCALE ENVIRONMENTS (SMILE. 1998, pp. 171–186.

[78] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Multi-view 3D Models from Single Images with a Convolutional Network". In: *European Conference on Computer Vision*. 2016, pp. 322–337.

[79] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. "What Do Single-View 3D Reconstruction Networks Learn?" In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3400–3409.

[80] *The Sims Resource 2015. ELECTRONIC ARTS.* `https://www.thesimsresource.com/`.

[81] Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. "Multi-view Consistency as Supervisory Signal for Learning Shape and Pose Prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[82] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. "Multi-View Supervision for Single-View Reconstruction via Differentiable Ray Consistency". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[83] Jinglu Wang, Bo Sun, and Yan Lu. "Mvpnet: Multi-view Point Regression Networks for 3D Object Reconstruction from a Single Image". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 8949–8956.

[84] Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. "Example-based hair geometry synthesis". In: *ACM SIGGRAPH 2009 papers*. 2009, pp. 1–9.

[85] Nan Wang, Haizhou Ai, and Shihong Lao. “A Compositional Exemplar-Based Model for Hair Segmentation”. In: *Proceedings of the 10th Asian Conference on Computer Vision*. 2010, pp. 171–184.

[86] Nan Wang, Haizhou Ai, and Feng Tang. “What Are Good Parts for Hair Shape Modeling?” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 662–669.

[87] Kelly Ward, Florence Bertails, Tae-Yong Kim, Stephen R. Marschner, Marie-Paule Cani, and Ming C. Lin. “A Survey on Hair Modeling: Styling, Simulation, and Rendering”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.2 (2007), pp. 213–234.

[88] Yichen Wei, Eyal Ofek, Long Quan, and Heung-Yeung Shum. “Modeling Hair from Multiple Views”. In: *ACM Transactions on Graphics*. 2005, pp. 816–820.

[89] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. “MarrNet: 3D Shape Reconstruction via 2.5D Sketches”. In: *Advances In Neural Information Processing Systems*. Vol. 30. 2017.

[90] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. “3D ShapeNets: A Deep Representation for Volumetric Shapes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1912–1920.

[91] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. “Objectnet3d: A Large Scale Database for 3D Object Recognition”. In: *Proceedings of the European Conference on Computer Vision*. Springer. 2016, pp. 160–176.

[92] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild". In: *IEEE Winter Conference on Applications of Computer Vision*. 2014, pp. 75–82.

[93] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. "Pix2Vox: Context-Aware 3D Reconstruction From Single and Multi-View Images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2690–2698.

[94] Jun Xing, Koki Nagano, Weikai Chen, Haotian Xu, Li-yi Wei, Yajie Zhao, Jingwan Lu, Byungmoon Kim, and Hao Li. "HairBrush for Immersive Data-Driven Hair Modeling". In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 2019, pp. 263–279.

[95] Yaser Yacoob and Larry S. Davis. "Detection and Analysis of Hair". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.7 (2006), pp. 1164–1169.

[96] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. "Dense 3D Object Reconstruction from a Single Depth View". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.12 (2019), pp. 2820–2834.

[97] Lingchen Yang, Zefeng Shi, Youyi Zheng, and Kun Zhou. "Dynamic Hair Modeling from Monocular Videos Using Deep Neural Networks". In: *ACM Transactions on Graphics* 38.6 (2019).

[98] Meng Zhang, Menglei Chai, Hongzhi Wu, Hao Yang, and Kun Zhou. "A Data-Driven Approach to Four-View Image-Based Hair Modeling". In: *ACM Transactions on Graphics*. 36.4 (2017).

[99] Meng Zhang and Youyi Zheng. “Hair-GAN: Recovering 3D Hair Structure from a Single Image using Generative Adversarial Networks”. In: *Visual Informatics* 3.2 (2019), pp. 102–112.

[100] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. “Shape from Shading: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.8 (1999), pp. 690–706.

[101] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. “Pyramid Scene Parsing Network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[102] Yi Zhou, Liwen Hu, Jun Xing, Weikai Chen, Han-Wei Kung, Xin Tong, and Hao Li. “Hairnet: Single-view Hair Reconstruction using Convolutional Neural Networks”. In: *Proceedings of the European Conference on Computer Vision*. 2018, pp. 235–251.

[103] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. “3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks”. In: *2017 IEEE International Conference on Computer Vision*. 2017, pp. 900–909.