GroomGen: A High-Quality Generative Hair Model Using Hierarchical Latent Representations

YUXIAO ZHOU, ETH Zurich, Switzerland
MENGLEI CHAI, Google Inc., United States of America
ALESSANDRO PEPE, Google Inc., United States of America
MARKUS GROSS, ETH Zurich, Switzerland
THABO BEELER, Google Inc., Switzerland

Fig. 1. Our method is able to automatically generate diverse high-quality hairstyles from random latent vectors.

Despite recent successes in hair acquisition that fits a high-dimensional hair model to a specific input subject, generative hair models, which establish general embedding spaces for encoding, editing, and sampling diverse hairstyles, are way less explored. In this paper, we present *GroomGen*, the first generative model designed for hair geometry composed of highly-detailed dense strands. Our approach is motivated by two key ideas. First, we construct *hair latent spaces* covering both individual strands and hairstyles. The latent spaces are compact, expressive, and well-constrained for high-quality and diverse sampling. Second, we adopt a *hierarchical hair representation* that parameterizes a complete hair model to three levels: single strands, sparse guide hairs, and complete dense hairs. This representation is critical to the compactness of latent spaces, the robustness of training, and the efficiency of inference. Based on this hierarchical latent representation, our proposed pipeline consists of a *strand-VAE* and a *hairstyle-VAE* that encode an individual strand and a set of guide hairs to their respective latent spaces, and a *hybrid densification step* that populates sparse guide hairs to a dense hair model. *GroomGen* not only enables novel hairstyle sampling and plausible hairstyle interpolation, but also supports interactive editing of complex hairstyles, or can serve as strong data-driven prior for hairstyle reconstruction from images. We demonstrate the superiority of our approach with qualitative examples of diverse sampled hairstyles and quantitative evaluation of generation quality regarding every single component and the entire pipeline.

CCS Concepts: • **Computing methodologies → Parametric curve and surface models**.

Additional Key Words and Phrases: Strand-level hair modeling, hairstyle generation

ACM Reference Format:
Yuxiao Zhou, Menglei Chai, Alessandro Pepe, Markus Gross, and Thabo Beeler. 2023. GroomGen: A High-Quality Generative Hair Model Using Hierarchical Latent Representations. *ACM Trans. Graph.* 42, 6, Article 267 (December 2023), 16 pages. https://doi.org/10.1145/3618309

Authors' addresses: Yuxiao Zhou, ETH Zurich, Switzerland, yuxiao.zhou@inf.ethz.ch; Menglei Chai, Google Inc., United States of America, mengleichai@google.com; Alessandro Pepe, Google Inc., United States of America, apepe@google.com; Markus Gross, ETH Zurich, Switzerland, grossm@inf.ethz.ch; Thabo Beeler, Google Inc., Switzerland, tbeeler@google.com.

0730-0301/2023/12-ART267
https://doi.org/10.1145/3618309

1 INTRODUCTION

Hair substantially contributes to a person's appearance, and we frequently change it to express ourselves. As such it plays a critical role in depicting not just our physical appearance but also reflecting our individuality, mood, and cultural belonging. Hair digitization and modeling have recently garnered much attention, highlighting the exciting potential of creating high-quality hairstyles that contribute significantly to the perceived realism of virtual human avatars. However, unlike any other parts of ourselves, such as faces,

bodies, or hands, hair geometry is highly intricate and unstructured, making it exceptionally challenging to represent or model.

Empirically, the complexity of hair arises from two main levels. Locally, each individual strand corresponds to a 1D curve embedded in 3D space, originating from the scalp, defined by intrinsic properties that give rise to diverse curliness or waviness, and modulated by external conditions such as gravity. Globally, comprising hundreds of thousands of hair strands, the overall hairstyle exhibits a high-level structure that combines coherency among neighboring strands with independent variations on a per-strand basis. Given the intricate nature of hair, much of the existing research focuses on hair *acquisition*, which often involves overfitting high-dimensional hair representations, typically Euclidean positions of densely sampled vertices, to various types of inputs (multi-view images [Beeler et al. 2012; Luo et al. 2012; Nam et al. 2019; Winberg et al. 2022; Zhang et al. 2017], single photos [Chai et al. 2016; Hu et al. 2015], or specialized sensors [Herrera et al. 2012]). While high-fidelity reconstruction for specific subjects is achieved, without proper parameterization that embeds the different hair within a shared compact space, their outputs lack the generalization capability necessary to support interpolation, manipulation, or novel hairstyle synthesis.

On the other hand, *generative hair models*, the main focus of this work, remain relatively unexplored. The pioneering work on hair geometry synthesis [Wang et al. 2009] proposes a 2D hair embedding that can generate new hairstyles from given exemplars through texture synthesis. However, due to the inherent limitations of explicit strand geometry encoding and homogeneous texture synthesis, this approach can only handle short to medium-length hair with uniform styles. More recently, Volumetric Hair VAE [Saito et al. 2018] demonstrates the potential of generating novel hairstyles by interpolating existing ones in a latent volume space. However, the combination of volumetric flow field and post-processing strand tracing often results in over-smoothed geometry with limited strand-level detail and inter-strand variation. The objective of our work is to develop a novel architecture for strand-level hair generation, capable of synthesizing diverse hairstyles with high-quality dense geometry in a computation- and memory-efficient manner. To this end, we aim to establish a new hair representation that is highly compact and efficient, capable of capturing common hairstyles through a shared parameterization, and expressive enough to encompass both global structural characteristics and local fine details.

We introduce *GroomGen*, a generative model for diverse and high-quality hairstyle synthesis. Our method is rooted in a hierarchical hair representation, inspired by the conventional practice of guide-hair-based authoring in visual effects. We represent a hairstyle using three levels of abstraction: strand latent codes for *individual strands*, low-resolution latent-maps for *sparse guide hairs*, and high-resolution strand-maps for *dense hairstyles*. This hierarchy not only achieves significant compression, robust training, and efficient inference without compromising expressiveness, but also establishes a versatile multi-level embedding space for sampling diverse and valid hairstyles. Based on this hierarchical representation, we design the entire hair generation pipeline as three major components:

(1) At the single strand level, we employ a *strand variational autoencoder* (strand-VAE) to establish a low-dimensional latent space for encoding diverse strand geometry.
(2) Building upon the strand latent space, our *hairstyle variational autoencoder* (hairstyle-VAE) encodes the sparsely-sampled guide hairs into a hairstyle feature vector.
(3) To generate dense hair from sparse guide strands, we propose a GAN-based *neural upsampler* that synthesizes high-resolution hair geometry, followed by a heuristic refinement step that allows user control for customizing details.

Our compact model possesses the capability to represent and generate diverse hairstyles with high visual fidelity. This versatility makes it useful for a wide range of applications, such as simulation, generating training data for downstream models through (un)conditional sampling, serving as a powerful prior for robust image-based hair reconstruction, and facilitating rapid hairstyle creation and exploration for artists.

2 RELATED WORK

Strand Representation of Hairs. A common approach to represent a hair model is by using a collection of hair strands, where each strand is defined as a polyline consisting of tens or hundreds of vertices. While this representation is intuitive and expressive, it often becomes heavy and redundant. To address this issue, previous works such as [Bertails et al. 2006, 2005] propose to use the super-helix as a compact approximation of hair strands. This representation requires only a few parameters per strand, but the resulting reconstruction is typically over-smoothed. In the recent work by [Rosu et al. 2022], neural representations of hair strands are explored. The authors adopt the modulated sine network structure [Mehta et al. 2021] to map a strand into a low-dimensional latent space, achieving superior reconstruction results. In organizing the strands of a hair model, many previous works [Lyu et al. 2022; Rosu et al. 2022; Zhou et al. 2018] opt to parameterize the scalp area using UV unwrapping and assign strands to corresponding pixels. While such a UV-mapping representation facilitates the exploitation of spatial adjacency among strands, due to the huge number of hair strands, a high-resolution UV map is often required, resulting in significant computational costs. Taking advantage of the local similarity of human hairs, other works [Chai et al. 2014, 2017; Guan et al. 2012] propose to use a set of sparse guide strands as proxies for all hairs, leading to more efficient simulation, which is a common practice in the industry. In this paper, we choose to utilize the strand-based representation for its fidelity and flexibility. We follow the convention of using guide hairs to represent hairstyles, which helps reduce the computational and memory overhead compared to representing individual strands directly.

Volumetric Representation of Hairs. An alternative approach to representing a hair model is through volumetric representation, where the entire hair volume is voxelized, and each voxel contains information about the growth direction and other properties of the hairs within it. This voxelized representation often organizes the free-growing hairs into regular groups, making the hair structure easier to capture. Although impressive results [Kuang et al. 2022;

Saito et al. 2018; Wang et al. 2022; Wu et al. 2022; Yang et al. 2019] have been achieved, the expressiveness of volumetric representations is inherently limited by the granularity of the discretization. The in-voxel fusion process inevitably leads to over-smoothed results, especially when dealing with curly hair or instances where hair strands cross each other within the same voxel. Furthermore, the use of volumetric representation can be computationally expensive, particularly when dealing with large volumes of long hairstyles.

Hair Acquisition. Hair capturing is an active and evolving research area. While a few previous works [Herrera et al. 2012; Jakob et al. 2009] seek to capture hair geometry with specialized devices, most existing methods for hair capturing rely on consumer-grade single- or multi-view cameras [Paris et al. 2004, 2008; Wei et al. 2005]. Static hair reconstruction techniques [Beeler et al. 2012; Chai et al. 2013, 2012; Luo et al. 2013; Nam et al. 2019; Olszewski et al. 2020; Sun et al. 2021; Winberg et al. 2022; Zheng et al. 2023] aim to reconstruct 2D and 3D hair curves based on single-view visual cues and multi-view correspondences. The work of [Shen et al. 2021] seeks to infer hair geometry from user sketches. In addition, dynamic capturing methods [Hu et al. 2017; Xu et al. 2014; Zhang et al. 2012] take temporal consistency into consideration. Although these model-free methods demonstrate impressive results, they tend to be fragile in challenging cases and unconstrained environments due to the thin strand geometry, occlusion between strands, and complex hairstyle configurations. To enhance robustness, recent works have introduced prior models as constraints. Some approaches [Chai et al. 2015; Hu et al. 2014] employ geometric primitives as constraints for individual hairs during capturing, while others utilize hairstyle databases [Chai et al. 2016; Hu et al. 2015; Liang et al. 2018] for initialization and guidance, largely improving reconstruction quality and stability. Furthermore, with the advancements in deep learning, neural approaches have emerged as the new state-of-the-art. For instance, the work of [Zhou et al. 2018] captures hair from monocular images by extracting hairstyle features using convolution neural networks. The work of [Rosu et al. 2022] combines multi-view reconstruction with neural descriptors to achieve photorealistic telepresence. In addition to strand-based representations, volumetric methods [Kuang et al. 2022; Luo et al. 2012; Saito et al. 2018; Wu et al. 2022; Yang et al. 2019] also have made significant progress in capturing both static and dynamic hair. These works aim to provide reliable priors for robust and high-quality hair acquisition, presenting a potential application of our research.

Hair Generation. Compared to hair acquisition, generative hair models are relatively unexplored. A heuristic example-based method for hair generation is proposed by [Ren et al. 2021], which is largely limited by the reference hair models. Variational autoencoder (VAE) [Kingma and Welling 2014] is widely recognized as one prevalent generative architecture. In the context of hair modeling with volumetric representation, the work of [Saito et al. 2018] proposes the utilization of VAEs. However, VAEs often suffer from over-smoothness issues despite their extensive examination for data embedding. Generative adversarial networks (GAN) [Goodfellow et al. 2014] have also demonstrated remarkable results in image generation [Karras et al. 2019; Radford et al. 2016]. By incorporating the perceptual discriminator loss, GANs are particularly effective in recovering fine detail with weak supervision. For our specific task, we employ both VAEs and GANs, where two VAE models encode individual strands and overall hairstyles, while another GAN model is adopted for detail restoration.

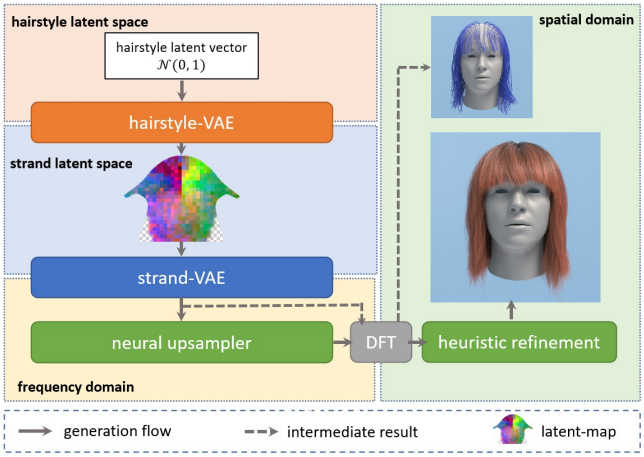

Fig. 2. We build a hierarchical representation of human hairstyles. To generate a hair model, we first draw a random vector from the Normal distribution in the hairstyle latent space. The vector is decoded by the hairstyle-VAE to get a low-resolution latent-map, which corresponds to sparse guide strands. Finally, the neural upsampler synthesizes dense hair strands from the sparse guide strands, which are further refined heuristically based on user specification.

3 METHOD

In this section, we present our comprehensive pipeline for hairstyle generation, tailored specifically for strand-based hair models. Our algorithm operates on three hierarchical levels of human hair: individual strands, sparse guide strands, and the complete hair model with dense hair strands. Accordingly, we design three components for each hierarchical level in hairstyle generation: 1) At the single strand level, our *strand variational autoencoder* (*strand-VAE*) establishes a low-dimensional latent space for encoding strands (Sec. 3.1). 2) Building upon the strand latent space, a *hairstyle variational autoencoder* (*hairstyle-VAE*) further encodes a hair model, represented by a collection of sparsely-sampled guide strands, into a feature vector (Sec. 3.3). 3) A hybrid densification step consisting of a neural upsampler (Sec. 3.4) and a heuristic refiner (Sec. 3.5) synthesizes the full hair style from the sparse guides. By connecting all these components together, our pipeline provides a complete framework for hair model generation. The overall structure of the pipeline is illustrated in Fig. 2.

3.1 Strand Latent Space

Our strand-VAE module performs the encoding of individual strands, transforming their Euclidean coordinates into a low-dimensional strand-wise latent space. This latent space serves as the foundation for guide strand generation and quasi-static simulation. Compared to raw Euclidean coordinates, our latent encoding is better regularized to a more constrained distribution of valid strands, without

sacrificing strand-level geometric details, coherency among strands, or the overall diversity of the generated hairstyle.

Typically, a strand is represented as a polyline consisting of N_s uniformly sampled points. Previous work [Rosu et al. 2022] constructs a latent space based on this polyline representation. However, we observe that representing strands in the original Euclidean space often has a significant negative impact on the structure preservation of the resulting latent space, leading to over-smoothed hair generation. To better retain strand-level details like curliness, we parameterize the strands in the frequency domain using the discrete Fourier transform (DFT), and establish the strand latent space based on this frequency representation. As elaborated in Sec. 4.3, this frequency latent space aligns better with human perception compared to the spatial latent space.

Formally, in the Euclidean spatial domain, a strand is originally represented as a polyline with N_s points: $\mathcal{S} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{N_s}\} \in \mathbb{R}^{N_s \times 3}$ ($N_s = 100$ in our case). We first compute the gradients as $\tilde{\mathcal{S}} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_{N_s-1}\} \in \mathbb{R}^{(N_s-1)\times 3}$ with the gradient displacement $\boldsymbol{d}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$, and then equally divide the entire strand into N_g non-overlapping segments: $\tilde{\mathcal{S}}_i = \{\boldsymbol{d}_{ik}, \boldsymbol{d}_{ik+1}, \ldots, \boldsymbol{d}_{ik+k-1}\}, i \in \{1, 2, ..., N_g\}$ with $k = \lceil (N_s - 1)/N_g \rceil$ denoting the segment size. The strands are segmented to allow for varying shape statistics along the strand. In all our experiments, we set $N_g = 3$. For each segment, we apply the DFT along x, y, z axes separately with respect to vertex indices, obtaining coefficients of Fourier bases $\mathcal{F}_i^a \in \mathbb{C}^f$, with $a \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$ referring to the axes and $f = \lfloor k/2 \rfloor + 1$ representing the number of frequency bands. Instead of using $\mathcal{F}$ directly, we further decompose it into three parts by taking their physical meanings into consideration: $\mathcal{F}_A = \mathrm{abs}(\mathcal{F}) \in \mathbb{R}^f$, $\mathcal{F}_{\cos} = \mathrm{real}(\frac{\mathcal{F}}{\mathcal{F}_A}) \in \mathbb{R}^f$, and $\mathcal{F}_{\sin} = \mathrm{img}(\frac{\mathcal{F}}{\mathcal{F}_A}) \in \mathbb{R}^f$, where $\mathrm{abs}(\cdot)$, $\mathrm{real}(\cdot)$, and $\mathrm{img}(\cdot)$ refer to the absolute value, real part, and imaginary part of a complex number. Here, $\mathcal{F}_A$ describes the amplitude of each frequency, intuitively the significance of the strand's curliness and length; $\mathcal{F}_{\cos}$ and $\mathcal{F}_{\sin}$ together describe the phase of the curves. We encode the phase as vector $(\mathcal{F}_{\cos}, \mathcal{F}_{\sin})$ instead of a scalar phase angle to avoid issues with periodicity. Concatenating $\mathcal{F}_A$, $\mathcal{F}_{\cos}$, and $\mathcal{F}_{\sin}$ for all segments and axes, we obtain a vector $\mathcal{V}$, namely the *frequency code*, of $N_g \times f \times 3 \times 3 = 459$ dimensions, to represent a strand in the frequency domain.

Our strand-VAE takes $\mathcal{V}$ as both the input and reconstruction target. The employed training loss terms include: L1 loss $\mathcal{L}_{\mathrm{A}}$ for the amplitude coefficients $\mathcal{F}_{\mathrm{A}}$; L1 loss $\mathcal{L}_{\mathrm{P}}$ for the phase coefficients $\mathcal{F}_{\cos}$ and $\mathcal{F}_{\sin}$; and KL divergence loss $\mathcal{L}_{KL}^s$ with weight $\lambda_{KL}^s = 10^{-4}$:

$$\mathcal{L}_s = \mathcal{L}_{\mathrm{A}} + \mathcal{L}_{\mathrm{P}} + \lambda_{KL}^s \mathcal{L}_{KL}^s. \tag{1}$$

Since the phases of the high-amplitude components play a more crucial role, the phase loss $\mathcal{L}_{\mathrm{P}}$ on each frequency is weighted by the corresponding ground truth amplitude $\hat{\mathcal{F}}_{\mathrm{A}}$:

$$\mathcal{L}_{\mathrm{P}} = \sum_{i=1}^{f} \tilde{\mathcal{F}}_{\mathrm{A}}^i * (|\mathcal{F}_{\sin}^i - \hat{\mathcal{F}}_{\sin}^i| + |\mathcal{F}_{\cos}^i - \hat{\mathcal{F}}_{\cos}^i|), \tag{2}$$

$$\tilde{\mathcal{F}}_{\mathrm{A}}^i = \frac{\hat{\mathcal{F}}_{\mathrm{A}}^i}{\sum_{j=1}^{f} \hat{\mathcal{F}}_{\mathrm{A}}^j}. \tag{3}$$

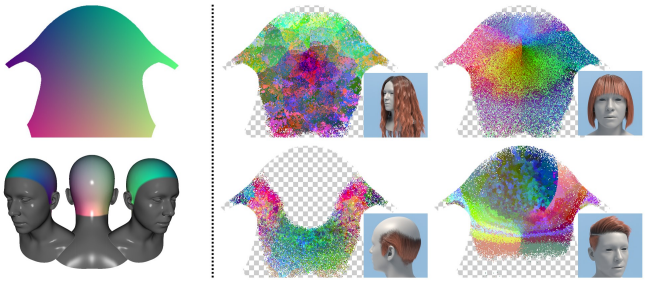

Fig. 3. Illustration of our scalp parameterization (left) and latent-maps with baldness (right). Visualized using selected 3 axes of the strand latent codes.

Here, $\hat{\cdot}$ denotes the ground truth values and $\tilde{\mathcal{F}}_{\mathrm{A}}^i$ represents the normalized weight for each frequency band i. The summation over the axes and segments is omitted here for conciseness.

The encoder of the strand-VAE consists of a fully-connected network with 7 layers. Except the input and output layers, each layer has 1024 hidden units with batch normalization [Ioffe and Szegedy 2015] and residual connection [He et al. 2016]. It takes the individual strand representation $\mathcal{V}$ as input and compresses it into a latent code $\boldsymbol{l} \in \mathbb{R}^{D_s}$. We use $D_s = 64$ in our experiments, resulting in a compression rate of 21.5%. On the other hand, the decoder follows the modulated sine network structure [Mehta et al. 2021] with 6 layers and 1024 hidden units. Given a latent code $\boldsymbol{l}$, the decoder generates the vector $\mathcal{V}$, which can then be converted back to Euclidean coordinates $\mathcal{S}$, with the additional input of the root position $\boldsymbol{p}_1$ pre-defined on the scalp.

3.2 Scalp Space Hairstyle Parameterization

We now introduce our representation of hairstyles. Similar to prior works [Lyu et al. 2022; Rosu et al. 2022; Wang et al. 2009; Zhou et al. 2018], we define the 2D parameterization of hairs on the scalp surface as a regular UV map, as illustrated in Fig. 3. The strand representations are embedded into the UV map at the positions corresponding to their roots on the scalp. When the strands are represented by frequency codes $\mathcal{V}$, the corresponding UV map is referred to as a *strand-map*; when the strands are represented by their latent codes, the UV map is called a *latent-map*. These two maps can be mutually converted using the strand-VAE.

In our pipeline, we employ two different resolutions for the latent-maps: 24×32 (referred to as a low-resolution map where 1 pixel has side length 1.0−2.9cm) and 216×288 (referred to as a high-resolution map where 1 pixel has side length 0.1 − 0.3cm). As not all texels are used, the low-resolution maps usually accommodate around 300 hairs, while the high-resolution ones contain around 25K hairs. Initially, we generate a low-resolution latent-map that corresponds to sparse guide strands, and then adopt a hybrid densification step to generate dense hair strands from the guide strands. This design choice is motivated by the observation of high redundancy in dense hairs due to the local coherency of nearby strands. Compared to directly using high-resolution latent-maps (256×256 in [Rosu et al. 2022] and 128×128 in [Lyu et al. 2022]), our intermediate representation enables better convergence during training and higher

computational efficiency during inference. This design also aligns with the common CG practice of using sparse guide strands to model and control the global hairstyle structure before densification.

Additionally, to ensure generalizability to a broader range of hairstyles, we incorporate baldness as part of the hairstyle. Baldness is defined by an additional binary mask, referred to as a *baldness-map* within the scalp space, as depicted in Fig. 3.

3.3 Hairstyle Latent Space

Based on the scalp space parameterization, our hairstyle-VAE learns to generate whole hairstyles utilizing the VAE framework. The input and reconstruction target for the hairstyle-VAE consist of both the low-resolution latent-map $\mathcal{M}_l \in \mathbb{R}^{w_M \times h_M \times D_s}$ and the baldness-map $\mathcal{M}_b \in \mathbb{R}^{w_M \times h_M}$, where w_M and h_M represent the width and height of both maps. Within the hairstyle-VAE, the encoder projects the latent-map $\mathcal{M} = \{\mathcal{M}_l, \mathcal{M}_b\}$ into a single latent vector $\boldsymbol{h} \in \mathbb{R}^{D_h}$ (we set $D_h = 512$, resulting in a compression rate of 99%), and the decoder takes the latent vector $\boldsymbol{h}$ as input to reconstruct the corresponding latent-map. The training objective is defined as:

$$\mathcal{L}_h = \mathcal{L}_{\text{rec}}^{\mathcal{M}_l} + \mathcal{L}_{\text{rec}}^{\mathcal{M}_b} + \lambda_{KL}^h \mathcal{L}_{KL}^h, \tag{4}$$

where $\mathcal{L}_{\text{rec}}^{\mathcal{M}_l}$ and $\mathcal{L}_{\text{rec}}^{\mathcal{M}_b}$ are the L1 reconstruction losses for latent-map $\mathcal{M}_l$ and baldness-map $\mathcal{M}_b$, and $\mathcal{L}_{KL}^h$ is the KL divergence loss with weight $\lambda_{KL}^h = 0.01$.

Our hairstyle-VAE utilizes a concise network architecture. The encoder part consists of a total of 12 convolutional layers, incorporating residual connections. Similarly, the decoder is symmetric to the encoder and employs transposed convolutions for upsampling. Please see Appendix D for detailed network structures.

3.4 Neural Upsampling

The hairstyle-VAE produces a low-resolution latent-map that represents sparse guide strands. To further generate a complete hair model with around 150K strands, we then employ a hybrid densification process involving two steps: upsampling and refinement. The upsampling step outputs a high-resolution strand-map with 25K hairs, and the refinement step additionally populates the strands by 6 times.

We emphasize that an end-to-end model is less suitable here because the mapping from sparse guide strands to dense hair strands is one-to-many, and the user's involvement is often necessary to resolve the ambiguity. Our two-step hybrid approach strikes a balance between simplicity and controllability. In the first step, our novel neural upsampler automatically populates the strands based on the guide strands. In the second step, users are allowed to refine the high-frequency details, providing control over the final results. In this section, we will introduce the first step of neural upsampling, while the second step will be elaborated in Sec. 3.5.

In the upsampling step, we aim to estimate a high-resolution strand-map from a low-resolution one where pixels contain frequency representations $\mathcal{F}$ of guide strands. The low-resolution strand-map is obtained by decoding the output of the hairstyle-VAE with the strand-VAE. Analogous to image upsampling, we assume that each dense strand can be viewed as a linear interpolation of its four neighboring guide strands. Consequently, the task simplifies

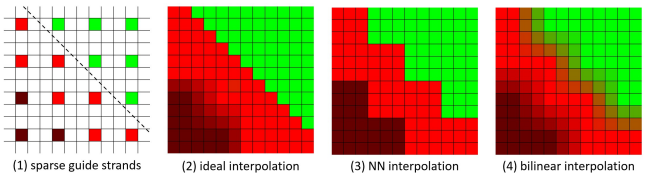

Fig. 4. Illustration of interpolating near a parting line. (1) shows the source low-resolution strand-map where red and green represent strands growing in opposite directions and the dashed line is the expected parting line. The ideal interpolation is (2) where the parting line is sharp while other regions are smooth. The nearest-neighbor interpolation in (3) has aliasing artifacts. The bilinear interpolation in (4) smooths out the parting line and the strands may even penetrate the head mesh.

to estimating the interpolation weights of the guide strands at each position. Trivial interpolation methods are inadequate for this task due to the significant variation in local smoothness observed in human hairstyles. Bilinear interpolation, for example, smooths out sharp parting lines and result in hair-head penetration, while nearest-neighbor interpolation exhibits aliasing artifacts (as depicted in Fig. 4). Therefore, spatially varying interpolation is often preferred, but determining an effective interpolation strategy *a priori* is challenging. While the industry relies on intensive manual design for this purpose, we introduce a *neural upsampler* to automate the process and ensure realism.

The input of the neural upsampler is a high-resolution multi-channel feature map that describes the guide strands distribution. At each pixel, the feature vector is formed by concatenating the low-frequency components of its four neighboring guide strands, their bilinear interpolation, and the distances to the guides. Only the low-frequency components are considered, with a cut-off frequency set to $f_l = 8$. This choice is made because the high-frequency details have less significance in the interpolation process and will be refined later on. The output of the neural upsampler is a 5-channel *weight-map*, where the first 4 channels represent the weights for the four neighboring guide strands, and the last channel represents the weight for the bilinear interpolation of the guide strands. At inference, each strand Y on the interpolated high-resolution map is computed as $Y = a_1X_1 + a_2X_2 + a_3X_3 + a_4X_4 + a_5\mathcal{B}(X_1, X_2, X_3, X_4)$, where $\{a_i\}$ represent the predicted weights, $\{X_i\}$ represent the neighboring guide hairs from the low-resolution map, and $\mathcal{B}(\cdot)$ denotes the bilinear interpolation of guide hairs at the position of Y. The bilinear interpolation is an effective shortcut because oftentimes it already provides a solution close to the optimal one.

The neural upsampler is trained in an adversarial framework (GAN) [Goodfellow et al. 2014] for two reasons: First, dense hair interpolation is not a deterministic task and there is no unique ground truth; Second, for dense hairstyles, it is more reasonable to perceptually evaluate the hair model as a whole rather than enforcing per-strand supervision. We devise the neural upsampler as a 12-layer convolutional network with residual connections and instance normalization [Ulyanov et al. 2016], and use a large kernel size of 13 to perceive more spatial information in this high-resolution setting (more details in Appendix D). The discriminator has the same structure as the generator, except that its input is the interpolated

high-resolution strand-map and its output is a score map. The loss function follows the Wasserstein loss [Arjovsky et al. 2017]. Denoting the neural upsampler as G and the discriminator as D, the loss function for the discriminator is:

$$\mathcal{L}_D = D(\mathcal{H}) - D(\mathcal{X}) + D(\mathcal{H})^2 + D(\mathcal{X})^2, \tag{5}$$

where $\mathcal{X}$ is a real strand-map from the dataset and $\mathcal{H}$ is a generated one. $D(\mathcal{H})^2$ and $D(\mathcal{X})^2$ are regularization terms that prevent $D(\mathcal{H})$ and $D(\mathcal{X})$ from being numerically too large. The loss function for the generator is:

$$\mathcal{L}_G = -D(\mathcal{H}) + \lambda_G(\mathcal{L}_G^{\text{bl}} + \mathcal{L}_G^{\text{g}} + \mathcal{L}_G^{\text{sum}}). \tag{6}$$

Denoting the 5-channel interpolation weights estimated by G as $\mathcal{W}$, $\mathcal{L}_G^{\text{bl}} = |\mathcal{W}_5 - 1|$ biases the weight of bilinear interpolation (channel 5) towards 1, $\mathcal{L}_G^{\text{g}} = \sum_{i=1}^{4} |\mathcal{W}_i|$ regularizes the weights of each guide towards 0, and $\mathcal{L}_G^{\text{sum}} = |\sum_{i=1}^{5} \mathcal{W}_i - 1|$ softly normalizes the weights. The regularization weight λ_G is set to 0.1.

3.5 Heuristic Refinement

The high-resolution map generated by the neural upsampler contains 25K strands, which is still fewer than normal human hairs. To further enhance the quantity and quality of the dense strands, we introduce a heuristic refinement step that increases the number of hairs to 150K with fine details. In this step, we provide the user with creative semantic control over the final appearance. This step can also be fully automatic with fixed or randomized parameters if a hands-off approach is preferred, e.g. for large scale data generation.

We start by addressing penetrations (detailed in Appendix C) and perturbing the *frequency representation* $\mathcal{F}$ with random noise to increase variation. The scale of the noise can be specified by the user, allowing for the creation of regular or messy hairstyles. Next, we perform wisp formation in the Euclidean spatial domain. The user may specify two parameters: the number of wisps w and the stickiness s to control the clustering of hairs. We adopt k-means clustering to identify w wisp clusters, and then guide each strand towards the center of its corresponding cluster:

$$\delta_i = \frac{s \cdot \min(1, \frac{l_i}{\bar{l}})}{\max(1, d_i^2)} + \sum_{k=1}^{i-1} \delta_k. \tag{7}$$

We denote the vertex index as i, where $i = 1$ is the fixed strand root with displacement $\delta_1 = \mathbf{0}$, and the deformation δ_i of other vertices is determined by stickiness s, distance to the center strand d_i, and length to the strand root l_i. To prevent excessive deformation near the root, we empirically set $\bar{l}$ to 5cm as a threshold. This simple deformation strategy can yield practically satisfactory results since the neural upsampler provides a good initialization. This wisp formation is skipped when $w = 0$ or $s = 0$. Finally, as the raw output of the neural upsampler only contains $25K$ strands, we duplicate all strands 6 times with small variations in the frequency domain again. This allows us to increase the total strand number of the final model to $150K$, further enhancing its density and realism.

4 EXPERIMENTS AND APPLICATIONS

Extensive experiments are conducted to validate the effectiveness of our hair generation pipeline. In Sec. 4.1, we introduce the dataset, training procedure, and system runtime. In Sec. 4.2, we evaluate each main component of our method. We present ablation studies to justify the major technical choices in Sec. 4.3. Finally, in Sec. 4.4, we introduce a quasi-static neural hair simulator as one downstream application of our model.

4.1 Datasets, Training, and Runtime

The model is trained and evaluated on an artist-created hairstyle dataset, referred to as GroomHair, which comprises diverse hairstyles with fine-grained variations. To create the dataset, the artists first identified 35 base hairstyle categories, encompassing a wide range of styles such as buzz, bobby, pixie, wavy, afro, and more (see Appendix F for the complete list). For each category, the artists utilize *Houdini*[*] to create a recipe that defines the desired hairstyles and generate a series of fine-grained variations (50 – 400 depending on the hairstyle) of the same category using different parameters. In some categories, baldness is also modeled, which corresponds to the baldness-map used by the hairstyle-VAE. The final dataset contains 7712 data samples, each representing a specific hair model with approximately 150K strands. We randomly split the entire dataset into 6940 training samples and 772 test samples. The training samples are further augmented by horizontal mirroring.

We first train our strand-VAE model using the GroomHair dataset. Subsequently, we fix the strand-VAE model and use it to process the hair models in GroomHair to obtain the training and testing data for the hairstyle-VAE.

The strand-VAE and hairstyle-VAE are both trained using the Adam [Kingma and Ba 2015] optimizer with an initial learning rate of 10^{-3}, which is reduced by a factor of 0.1 whenever the training loss ceases to improve. The training process continues until the learning rate reaches 10^{-6}. The neural upsampler is also trained using the Adam optimizer but with a fixed learning rate of 10^{-4} for a total of 105K iterations (around 27 epochs).

We evaluate the runtime performance of each module on a PC equipped with an Intel Core i9-11900KF CPU and an NVIDIA A100 40GB GPU. The results are summarized in Tab. 1. It is noteworthy that the runtime measurements for the strand-VAE and hairstyle-VAE only include the decoder parts of the networks. Our system exhibits real-time performance, achieving more than 30 FPS for the generation and simulation of 300 – 500 guide strands, which is a common quantity in hairstyle authoring. Users can edit the hairstyle by either tweaking the hairstyle code in the latent space or modifying the guide strands in the Euclidean space interactively. The heuristic refinement step typically takes 10 – 15 seconds, and the optional penetration correction step takes 30 – 55 seconds.

4.2 Evaluation

In this section, we provide a comprehensive assessment of the components of our system in a bottom-up order. First, we evaluate the performance of the strand-VAE in encoding individual strands into the strand latent space. Next, we assess the hairstyle-VAE in embedding a given hair model into the hairstyle latent space, as well as generating hair models from this latent space. Finally, we

[*]https://www.sidefx.com/products/houdini/

Table 1. Runtime and number of parameters of the modules. Our system achieves real-time performance for generation, editing, and simulating up to 500 strands before densification.

	component runtime (ms)				# params.
# strands	1	300	500	150K	
strand-VAE	1.90	14.9	23.5	7670	10.59M
neural simulator	3.26	4.58	4.77	1330	32.74M
hairstyle-VAE		3.42			83.65M
neural upsampler		281			11.22M

Table 2. Quantitative metrics of the strand-VAE, hairstyle-VAE, and neural simulator. The errors are acceptably low.

	strand-VAE	hairstyle-VAE	neural simulator
pos. err.	1.90mm	7.26mm	8.89mm
loc. err.	0.15mm	0.40mm	0.44mm

demonstrate how the densification step effectively produces realistic dense hairs from sparse guide strands.

We use the following per-strand metrics. Recall that originally each hair strand is represented as a polyline $\mathcal{S} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{N_s}\}$ and the parent-relative displacement is defined as $\boldsymbol{d}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$. Positional error (pos. err.) calculates the mean distance between corresponding points of the predicted strands and the ground truth (indicated by $\hat{\cdot}$): $\sum_i^{N_s} \|\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i\|/N_s$. Local position error (loc. err.) measures the L2 distance between the gradients of corresponding points on the strands, without accumulating errors along the hair: $\sum_i^{N_s-1} \|\boldsymbol{d}_i - \hat{\boldsymbol{d}}_i\|/(N_s - 1)$. We report these metrics by averaging them per hair model and then across the entire test set. This ensures that each hair model contributes equally to the final numbers.

Strand-VAE. We first evaluate the strand-VAE model on the test set by measuring the reconstruction error of encoding and decoding individual strands. The quantitative results are reported in Tab. 2 (1st column). The mean reconstruction error is remarkably low, measuring only 1.90mm. In Fig. 5 we provide a few examples of strand reconstruction from the test set. As demonstrated, the reconstruction is of high fidelity and the difference is hard to discern. These results indicate that the strand-VAE effectively constructs a latent space that serves as a solid foundation for subsequent steps.

Hairstyle-VAE. Next, We evaluate the performance of the hairstyle-VAE model by encoding and decoding the entire hair model represented as a latent-map. The reconstructed latent-map is then decoded by the strand-VAE to obtain hair strands in the Euclidean space for visualization and error computation. The resulting reconstruction errors are reported in Tab. 2 (2nd column). As encoding the entire hair model is generally more challenging, particularly in our setting where the hairstyle latent vector utilizes only 0.4% parameters of the guide hairs, we observe relatively larger errors compared to the strand-VAE. Nevertheless, the average error of 7mm remains at an acceptably low level, preserving the overall visual appearance of the hairstyle well as can be seen in Fig. 6, even for challenging hairstyles (right column) that have not been explored

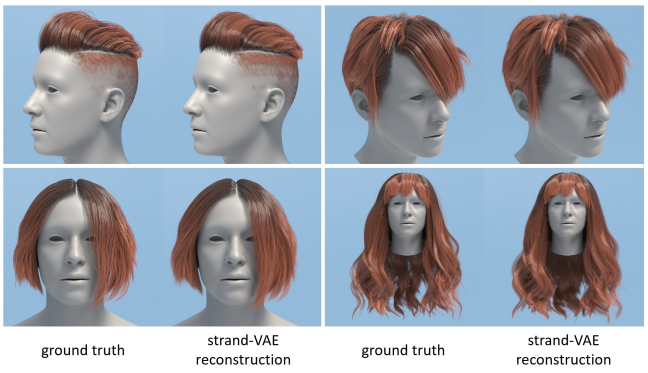

Fig. 5. Reconstruction results of strand-VAE on the test set by encoding and decoding each strand. The difference is barely observable. The average positional errors for the demonstrated samples are (left to right, top to bottom): 1.41mm, 1.13mm, 1.30mm, and 5.08mm.

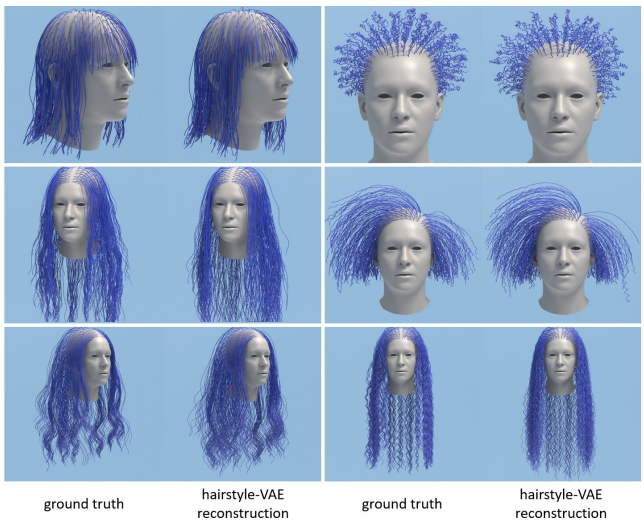

Fig. 6. Selected results of our hairstyle-VAE on the test set. Hairstyles are well-preserved with only 0.4% parameters of the original guide hairs. The average positional error for the demonstrated samples are (left to right, top to bottom): 8.35mm, 5.65mm, 24.53mm, 12.76mm, 24.61mm, and 20.68mm.

in previous works. Additionally, our model accurately reconstructs the baldness-map, a component often overlooked in prior work. The intersection-over-union (IoU) of the baldness-map on the test set measures 97.3% when the threshold is set to 0.8.

We now present a series of experiments to showcase the capability of the hairstyle-VAE in hairstyle generation and authoring. Firstly, we demonstrate that the latent space of the hairstyle-VAE is sufficiently well-constrained to allow for meaningful hairstyle generation by direct random sampling from the Normal distribution, as shown in Fig. 7. It is worth emphasizing the high local diversity observed in the generated hair styles, where the strands deviate from each other frequently, in contrast to previous works where nearby strands tend to grow in parallel and become over-smoothed.

Fig. 7. Diverse hairstyles generated by our hairstyle-VAE from random vectors sampled in the hairstyle latent space. We would like to emphasize 1) the diversity of hairstyles, which comes from the powerful hairstyle-VAE; 2) high local variety of the hairs, which originates from the well-structured frequency strand latent space.

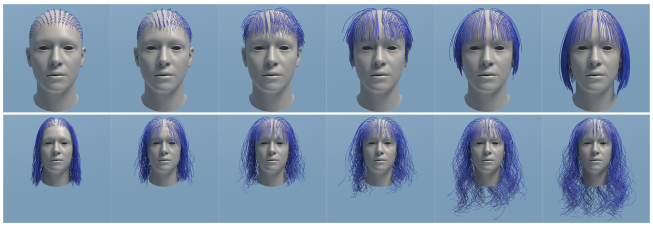

Fig. 8. Interpolation of hairstyles from left to right in the hairstyle latent space. While the start and end hairstyles are distinct, the interpolation trajectory is reasonable.

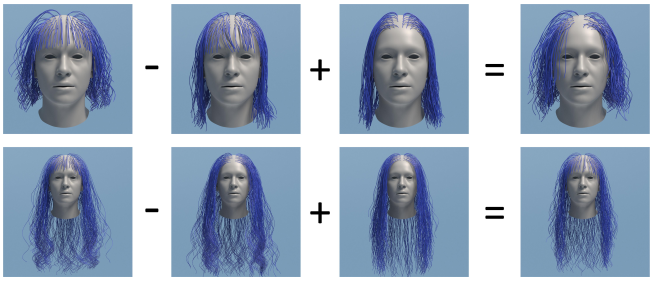

Fig. 9. Arithmetic between hairstyles: we subtract the hairstyle latent vector $\boldsymbol{h}$ of column 2 from column 1 and add the difference to column 3, finally get column 4. First row: the differences in curliness and length are successfully transferred. Second row: the difference between column 1 and 2 is the fringe, which is added to column 3 and gives us column 4. The hairstyles in the last column are *not* in the original dataset.

Additionally, we show that hairstyles can also be generated by traversing the latent space through interpolation, as shown in Fig. 8. Despite the distinct characteristics of the starting and ending hair models, the interpolation path in the hairstyle latent space remains semantically valid. This demonstrates the flexibility of our model in generating new hairstyles with controllable attributes.

Furthermore, we explore another interesting application of arithmetic operations between hairstyles, as shown in Fig. 9. Denoting the hairstyle latent vector of column i as $\boldsymbol{h}_i$, we compute $\boldsymbol{h}^* = \boldsymbol{h}_1 - \boldsymbol{h}_2 + \boldsymbol{h}_3$ and decode $\boldsymbol{h}^*$ to produce the resulting hair model. This simple arithmetic aligns well with human intuition and can generate novel hairstyles that do not exist in our dataset.

The most similar previous work is Volumetric Hair VAE (VHV) of [Saito et al. 2018] that learns the latent embedding of hair models from volumetric representation. However, a fair comparison is challenging due to fundamental differences in tasks (capture vs. generation), hair representations (voxels vs. strands), and datasets. For a rough comparison, we convert our predicted strands into volumetric representations and report the numbers in Tab. 3 (note that the test sets differ). The metrics of IoU, precision, and recall evaluate the correctness of strand occupancy in the space. While our latent space is much more compact than that in VHV, our method still outperforms VHV on these metrics. It is noteworthy that our method exhibits higher error on the growing flows due to the fact that the volumetric representation in VHV fuses growing directions within the same voxel, which conflicts with our strand representation without any fusion. Furthermore, our model can generate hairstyles simply from random sampling, while VHV only demonstrates interpolation of given similar hairstyles. Please find qualitative comparisons with VHV in Appendix E.

Hybrid Densification. In the densification step, we utilize our neural upsampler to increase the number of strands by upsampling low-resolution latent-maps to higher resolution. This is followed by

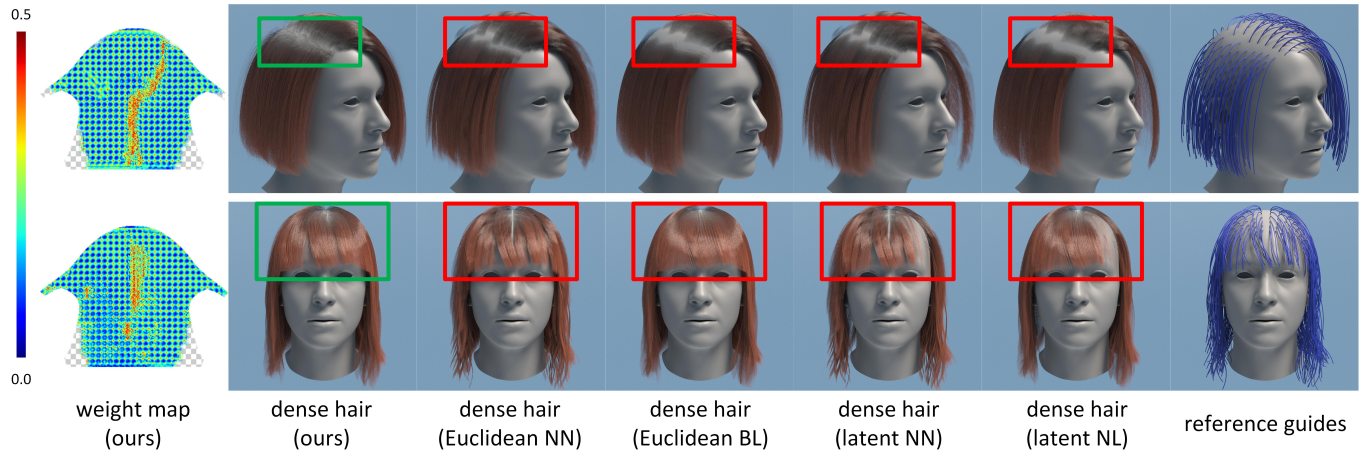

Fig. 10. Output from our neural upsampler based on our hairstyle-VAE reconstruction. Our method produces the most natural results, and the weight maps (please refer to 4.2 for an explanation) reveal the parting lines correctly. Nearest neighbor (NN) interpolation shows unrealistic abrupt changes between patches. Bilinear (BL) interpolation has severe hair-head penetration issues near the parting line.

Fig. 11. Our results after refinement. Each group is produced from the same output of the neural upsampler. The parameters effectively control the fine details without losing realism. Note the complex wisps structures, e.g. row 1 column 5, originates from the generated guide strands with high local variety.

Table 3. A **rough** comparison with VHV [Saito et al. 2018]. Our method has higher IoU, precision, and recall, which suggests better estimation of strand occupancy in the space. Our method has a slightly higher L2 flow error because we use a strand-based representation. Notably, ours has a much more compact latent space that supports direct random sampling.

	IoU	Precision	Recall	L2 (flow)	latent dim.
VHV	0.8243	0.8888	0.9191	0.2118mm	6144
ours	0.9426	0.9777	0.9626	0.2807mm	512

a heuristic refinement process with user-defined parameters. Please note that all the results presented in this step are based on the hairstyle-VAE's reconstruction and not ground truth guide strands.

We first evaluate the neural upsampler. In Fig. 10, we show a few representative hair models produced by the neural upsampler and

Table 4. Hair-head penetration rate of different interpolation methods. Our GAN-based neural upsampler avoids the aliasing artifacts in nearest neighbor interpolation but also keeps the parting line sharp, as indicated by a very low penetration rate.

	ours	NN	BL	full sup.	latent pred.
rate	1.5‰	0.0‰	5.0‰	2.1‰	195.5‰

compare them with alternative methods including nearest-neighbor (NN) and bilinear (BL) interpolation in both Euclidean and strand latent spaces. To visualize the weight-maps intuitively, we factorize the weight for bilinear interpolation into individual guide strands and colorize each pixel based on the standard deviation (*std*) of the weights. Larger *std* values indicate sharp transitions, while smaller *std* values reflect smooth interpolations. The emerging grid-like

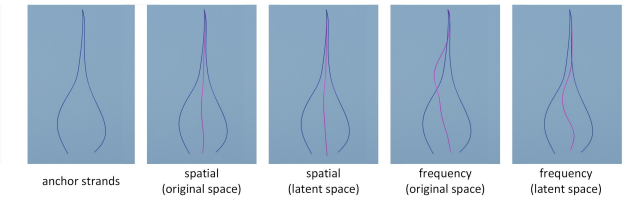

Fig. 12. Averaging two curly strands (column 1, blue strands) based on spatial representations (column 2 and 3, magenta strand) results in straight strands in both original space and latent space. While frequency representation (column 4, magenta strand) preserves the curliness, averaging in the latent space (column 5, magenta strand) gives the most natural result.

Table 5. Positional error (pos. err.) and relative messiness (rel. mes.) for hairstyle-VAE variants. The unit is millimeter. Our method has better reconstruction error and also similar levels of hair messiness as the ground truth data. Other models are either over-smooth or less accurate.

	latent space			original space	
	freq. (ours)	spat.	dense	spat.	freq.
pos. err.	7.25	7.08	9.80	7.07	8.47
rel. mes.	-0.05	-0.11	-0.23	-0.20	-0.08

structure illustrates how interpolation is inhibited when close to the guides and parting lines, while being smooth otherwise. To make artifacts more obvious, we remove all strands that penetrate the head mesh. In the first row of Fig. 10, our learned upsampler not only preserves the parting line but also avoids aliasing patterns observed in the NN interpolation (column 3 & 5). The presence of baldness in the bilinear interpolations (column 4 & 6) indicates severe penetrations when interpolating strands from opposite sides of the parting line. In the second row, our estimated parting line correctly ends before the fringe. Additionally, we report the percentage of hairs that penetrate the head mesh after upsampling in Tab. 4. Our neural upsampler stands out with a very low penetration rate, indicating that it accurately identifies most hairstyle parting lines. Please see Appendix A for more results of the neural upsampler.

In Fig. 11 we demonstrate the final results after the refinement step. Each group presents three hair models produced from the same output of the neural upsampler but with different user-defined parameters. Note that the intricate wisp structures, such as the one shown in the first row, fifth column, are rarely seen in previous works. The high fidelity of these structures is a result of our strand-level representation of hair models in the frequency latent space. Please find more results in Appendix A.

4.3 Ablation Study

In this section, we provide justifications for the important technical choices made in our approach. We first explain why our frequency representation of strands leads to a better latent space compared to conventional spatial coordinates. Then, we demonstrate the superiority of our GAN-based neural upsampler over other alternatives for strand-map upsampling. Lastly, we highlight the suitability of our hierarchical structure for representing a hair model.

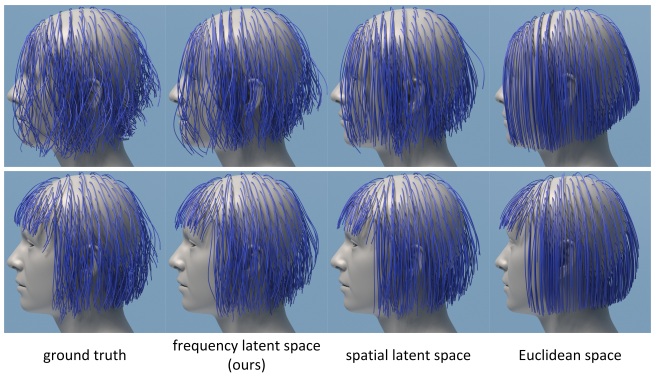

Fig. 13. The frequency hairstyle-VAE (ours) achieves significantly better local variety while other alternatives with different strand representations or latent spaces suffer from over-smoothness.

Strand VAE. Our method begins by constructing a compact latent space for strands, which effectively reduces the dimensionality while preserving high-fidelity shape information. Among various shape features, we consider length and curliness to be the most crucial ones. We opt to build the strand latent space in the frequency domain (*frequency latent space*), motivated by the fact that the Fourier spectrum explicitly encodes both features. In contrast, spatial coordinates do not directly represent curliness, so that the latent space thereon (*spatial latent space*) is not always consistent with respect to curliness. Consequently, two visually similar curly strands may be embedded far apart in this spatial latent space, while a straight strand and a curly strand may appear closer to each other. Such counter-intuitive issues are avoided in the frequency latent space. To validate this claim, we conduct additional experiments by training a separate strand-VAE model using spatial coordinates (*spatial strand-VAE*) and subsequently a hairstyle-VAE based on it (*spatial hairstyle-VAE*). We refer to our main models in the frequency domain as *frequency strand-VAE* and *frequency hairstyle-VAE*.

We first present an intuitive explanation for our reasoning in Fig. 12. We pick two curly hairs from our dataset and average them based on their spatial coordinates. The resulting strand appears straight and loses its curliness (column 2). Averaging in the spatial latent space leads to a similar outcome (column 3). In contrast, if the averaging is performed on the frequency code $\mathcal{V}$, the curliness is better preserved (column 4). The most meaningful result is obtained when the averaging is conducted in the frequency latent space (column 5). This is because, in the frequency domain, the averaging is applied independently to the amplitudes and phases. If both source strands have similar amplitudes but different phases, their mean will maintain the amplitude (i.e., curliness) while undergoing a phase shift. Consequently, the frequency latent space aligns better with the human perception of hair shapes, where curly and straight strands are distinguished regardless of their spatial proximity. Please find more results of strand interpolation in Appendix B.

This inherent structural distinction between the spatial and frequency strand latent spaces significantly influences the performance of the hairstyle-VAEs residing within them. This impact is shown in

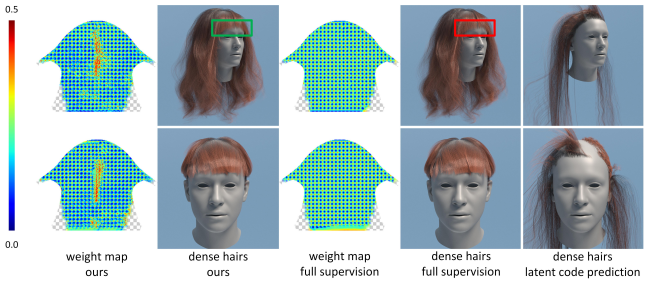

Fig. 14. Densification results from different upsamplers. Our method (column 1 & 2) works well on diverse hairstyles. Training with full supervision (column 3 & 4) is comparable to bilinear interpolation. Directly learning strand geometry fails to converge (column 5).

Fig. 13, where the frequency hairstyle-VAE (2nd column) exhibits a high level of local variety that closely resembles the ground truth, while the spatial hairstyle-VAE (3rd column) produces overly smooth results. In real hairstyles, it is quite common for adjacent strands to grow in opposite directions, resulting in interesting local variations. However, the embedding within the spatial domain tends to diminish such local variety, whereas the frequency domain preserves it better.

To quantitatively assess local variation, we introduce the *messiness* metric, defined as follows. For each strand i, we calculate its mean deviation relative to its neighbors:

$$\mathcal{D}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \frac{1}{N_s - 1} \sum_{k=1}^{N_s - 1} ||\boldsymbol{d}_{i,k} - \boldsymbol{d}_{j,k}||_2, \tag{8}$$

where $\mathcal{N}_i$ represents the neighbors of strand i, and $d_{i,k}$ denotes the parent-relative displacement of vertex k of strand i. The messiness metric is defined as the mean $\mathcal{D}_i$ of all strands, characterizing the uniformity of a hair model. A higher messiness value indicates greater local variety, while a lower value suggests a regular and smooth hairstyle. We report the difference in messiness relative to the ground truth dataset (0.428mm), where a lower value indicates a smoother result. As shown in Tab. 5 (rel. mes.), although our method produces a slightly more regular outcome compared to the ground truth, the spatial hairstyle-VAE exacerbates the gap, resulting in a doubling of the difference in messiness.

Notably, while positional encoding (PE) [Mildenhall et al. 2020] shares certain high-level similarities with discrete Fourier transform (DFT), they are substantially different in our context. PE considers the coordinates individually and expands each scalar value to a high-dimensional vector, while DFT considers a sequence and transforms it into the Fourier domain with the same dimension. For comparison, we train another strand-VAE based on PE and a corresponding hairstyle-VAE, and observe similar local over-smoothness (relative messiness: -0.11) as the vanilla spatial representation. This is because PE does not considers the strand shape as a whole. The average pos. error (strand-VAE: 1.70mm, hairstyle-VAE: 6.89mm) is also similar.

Neural Upsampler. The challenge of hair densification is two-fold. Firstly, the sparse guide strands only provide a general depiction of the overall hairstyle, making the mapping to dense hair strands indeterminate without unique ground truth. Secondly, humans perceive hairstyles holistically rather than focusing on individual strands. Per-strand supervision becomes impractical when dealing with as many as 25K strands. Therefore, we choose to adopt a GAN model for perceptual supervision.

To highlight the advantages of our GAN-based neural upsampler, we train an alternative model using full supervision. The ground truth data is generated by downsampling the hair models in our database. However, we find that this model cannot be effectively optimized and only converges to a local optimum that closely resembles bilinear interpolation. This occurs because the model attempts to learn a deterministic mapping that does not exist. The weight maps and qualitative results of this fully supervised model exhibit similar unnatural artifacts as bilinear interpolation and lack awareness of parting lines, as shown in Fig. 14 (column 3 & 4).

We choose to predict interpolation weights instead of strand geometry due to the greater constraints imposed on the weights. This leads to more stable training for the fragile GAN model. As an ablation analysis, we train another model that directly predicts strand latent codes using a discriminator loss. This model fails to converge to a valid solution and only generates meaningless strands, as shown in Fig. 14 (column 5) and Tab. 4 (column *latent pred.*).

Hierarchical Structure. Representing hairstyles with a hierarchical structure is one of our main design choices. We demonstrate the necessity of each level in the following analysis. At the strand level, encoding strands into a low-dimensional latent space effectively simplifies the generation task. To verify this, we conduct an ablation study by removing the strand-level abstraction. We train two alternative hairstyle-VAE models that directly take strand geometries from the original space. These models are denoted as *ori. spat.* (strands represented as spatial gradients $\bar{\mathcal{S}}$ in the original space) and *ori. freq.* (strands represented as frequency codes $\mathcal{V}$ in the original space). They share the same network structure as our standard hairstyle-VAE, except for different input and output dimensions. The quantitative evaluations are reported in Tab. 5. Although the ori. spat. variant achieves a slightly lower positional error, it suffers from over-smoothness, as shown in Fig. 13 (4th column). On the other hand, the ori. freq. variant produces a considerably worse reconstruction error, reported in Tab. 5.

At the hairstyle level, we utilize a set of sparse guide strands to describe a hair model instead of dense strands. Using guide strands as an intermediate descriptor is crucial for reducing the complexity of the task. In our case, dense strands are represented by a latent-map of shape $216 \times 288 \times 65$, containing more data than a high-resolution (1280×1024) RGB image. Compressing such a large amount of data into a low-dimensional vector poses a challenging task in its own right, let alone that no control is provided to the user in such an end-to-end model. For the ablation study, we remove the sparse guide strands level modeling and train an alternative hairstyle-VAE model that learns the mapping directly from dense strands to a single latent hairstyle vector, denoted as *dense* in Tab. 5. Despite having 1.6 times more parameters than the combined hairstyle-VAE and neural upsampler, this model still has worse reconstruction error and struggles with local over-smoothness.

Fig. 15. Our neural simulator gives plausible estimations for diverse hairstyles and head poses. It runs in real-time for up to 3K hairs, while conventional industrial simulators may take minutes.

4.4 Application: Quasi-Static Simulation

As an example application based on the latent representation of a hairstyle, we introduce a quasi-static simulator. Regarding the shape of a hairstyle with the straight head pose as the rest state, the neural simulator estimates hair deformation driven by different head poses. We model hair in the head coordinate frame, where the head is fixed while the direction of gravity can vary. Thus, the task is formulated as predicting strand deformation given a specific gravity direction.

Our quasi-static simulator is a neural network that takes as input a reference latent code $\boldsymbol{l}_r \in \mathbb{R}^{D_s}$ of the target strand under standard gravity ($-y$ direction), the position of the strand root $\boldsymbol{p}_1 \in \mathbb{R}^3$, and the new direction of gravity $\boldsymbol{h} \in \mathbb{R}^3$. The output is the latent code of the deformed strand. We design the network to be of 33 fully-connected layers with 1024 hidden units and residual connections, optimized using the L2 loss of the target latent codes. The training data is built from GroomHair where the head is randomly tilted and the equilibrium state of the guide strands is simulated using *Houdini*. The deformed strand geometries are embedded into the strand latent space with the pre-trained strand-VAE.

As shown in Tab. 2 the average error of the neural simulator is only 9mm. In Fig. 15, we provide qualitative results where the predicted deformation closely matches the ground truth. Compared to conventional methods, the neural simulator is faster by 3 magnitudes (Tab. 1), but still achieves satisfying quality, suggesting that the latent representation can support complex downstream tasks.

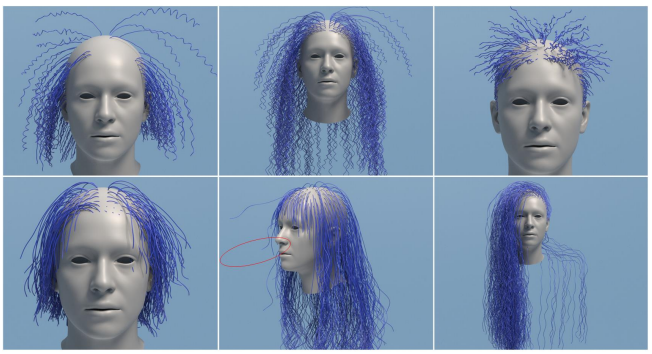

Fig. 16. Failure cases that are generated from random sampling. Top row: unnatural hairstyles do not conform to human aesthetics. Bottom row: physical artifacts such as penetration and flying long strands.

5 CONCLUSION

In this paper, we present the first generative hair model capable of automatically synthesizing diverse hairstyles. We demonstrate the effectiveness of embedding hairstyles into latent spaces with significantly fewer parameters through hierarchical decomposition. The strand latent space, based on frequency components, reduces dimensionality while preserving fidelity. The hairstyle latent space is well-constrained for generating guide strands. The neural upsampler effectively densifies guide strands into dense hairs, and the heuristic refinement process produces realistic final results with user control.

Limitations. First, the generation capability of our model is inherently bounded by the diversity of the dataset. The majority of the dataset comprises everyday hairstyles, while certain styles like braids are not adequately represented. Second, the entire system is trained on a specific head shape, embedded within the network weights. While the UV parameterization allows for adaptation of generated hair models to different head shapes to a certain extent, penetrations may still occur since no explicit information about the head mesh is provided to the system. Lastly, our current system does not explicitly consider physical attributes. The empirically devised wisp formation and penetration refinement steps lack a solid physical foundation, and the neural simulator infers hair deformation solely based on the rest shape and pre-defined gravity. In Fig. 16 we show a few failure cases from random generation.

Applications. The primary application of this work is automatic hairstyle generation. With the well-structured hairstyle latent space and the refinement steps, our method is capable of generating hair models that go beyond the training data to a certain extent. The hierarchical modularization of the pipeline also allows for human involvement, where artists can edit the generated guide strands before the automatic densification step or fine-tune the heuristic parameters for more precise control during the refinement stage. Additionally, we anticipate that the strand and hairstyle latent space can serve as reliable priors for hair geometry acquisition, leading to a hair capturing system that starts by optimizing the hairstyle and strand latent codes to fit the observed data.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments and suggestions, Denis Zen for modeling the hairstyles in the dataset, Shunsuke Saito for helping with the comparison, Xinyu Yi for proofreading, Daoye Wang for code review, and Erroll Wood and Chenglei Wu for fruitful discussions.

REFERENCES

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *ICML 2017*, Vol. 70. 214–223.

Thabo Beeler, Bernd Bickel, Gioacchino Noris, Paul A. Beardsley, Steve Marschner, Robert W. Sumner, and Markus H. Gross. 2012. Coupled 3D reconstruction of sparse facial hair and skin. *ACM Trans. Graph.* 31, 4 (2012), 117:1–117:10.

Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean Luc Lévêque. 2006. Super-helices for predicting the dynamics of natural hair. *ACM Trans. Graph.* 25, 3 (2006), 1180–1187.

Florence Bertails, Basile Audoly, Bernard Querleux, Frédéric Leroy, Jean Luc Lévêque, and Marie-Paule Cani. 2005. Predicting Natural Hair Shapes by Solving the Statics of Flexible Rods. In *Eurographics 2005*. 81–84.

Menglei Chai, Linjie Luo, Kalyan Sunkavalli, Nathan Carr, Sunil Hadap, and Kun Zhou. 2015. High-quality hair modeling from a single portrait photo. *ACM Trans. Graph.* 34, 6 (2015), 204:1–204:10.

Menglei Chai, Tianjia Shao, Hongzhi Wu, Yanlin Weng, and Kun Zhou. 2016. AutoHair: fully automatic hair modeling from a single image. *ACM Trans. Graph.* 35, 4 (2016), 116:1–116:12.

Menglei Chai, Lvdi Wang, Yanlin Weng, Xiaogang Jin, and Kun Zhou. 2013. Dynamic hair manipulation in images and videos. *ACM Trans. Graph.* 32, 4 (2013), 75:1–75:8.

Menglei Chai, Lvdi Wang, Yanlin Weng, Yizhou Yu, Baining Guo, and Kun Zhou. 2012. Single-view hair modeling for portrait manipulation. *ACM Trans. Graph.* 31, 4 (2012), 116:1–116:8.

Menglei Chai, Changxi Zheng, and Kun Zhou. 2014. A reduced model for interactive hairs. *ACM Trans. Graph.* 33, 4 (2014), 124:1–124:11.

Menglei Chai, Changxi Zheng, and Kun Zhou. 2017. Adaptive Skinning for Interactive Hair-Solid Simulation. *IEEE Trans. Vis. Comput. Graph.* 23, 7 (2017), 1725–1738.

Tamar Flash and Neville Hogan. 1985. The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. *Journal of Neuroscience* 5, 7 (1985), 1688–1703.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS 2014*. 2672–2680.

Peng Guan, Leonid Sigal, Valeria Reznitskaya, and Jessica K. Hodgins. 2012. Multi-linear Data-Driven Dynamic Hair Model with Efficient Hair-Body Collision Handling. In *SCA 2012*. 295–304.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR 2016*. 770–778.

Tomás Lay Herrera, Arno Zinke, and Andreas Weber. 2012. Lighting hair from the inside: a thermal approach to hair reconstruction. *ACM Trans. Graph.* 31, 6 (2012), 146:1–146:9.

Liwen Hu, Derek Bradley, Hao Li, and Thabo Beeler. 2017. Simulation-Ready Hair Capture. *Comput. Graph. Forum* 36, 2 (2017), 281–294.

Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. 2014. Robust hair capture using simulated examples. *ACM Trans. Graph.* 33, 4 (2014), 126:1–126:10.

Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. 2015. Single-view hair modeling using a hairstyle database. *ACM Trans. Graph.* 34, 4 (2015), 125:1–125:9.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML 2015*, Vol. 37. 448–456.

Wenzel Jakob, Jonathan T. Moon, and Steve Marschner. 2009. Capturing hair assemblies fiber by fiber. *ACM Trans. Graph.* 28, 5 (2009), 164.

Tero Karras, Samuli Laine, and Timo Aila. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. In *CVPR 2019*. 4401–4410.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR 2014*.

Zhiyi Kuang, Yiyang Chen, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2022. DeepMVSHair: Deep Hair Modeling from Sparse Views. In *SIGGRAPH Asia 2022*. 10:1–10:8.

Shu Liang, Xiufeng Huang, Xianyu Meng, Kunyao Chen, Linda G. Shapiro, and Ira Kemelmacher-Shlizerman. 2018. Video to fully automatic 3D hair model. *ACM Trans. Graph.* 37, 6 (2018), 206.

Linjie Luo, Hao Li, Sylvain Paris, Thibaut Weise, Mark Pauly, and Szymon Rusinkiewicz. 2012. Multi-view hair capture using orientation fields. In *CVPR 2012*. 1490–1497.

Linjie Luo, Hao Li, and Szymon Rusinkiewicz. 2013. Structure-aware hair capture. *ACM Trans. Graph.* 32, 4 (2013), 76:1–76:12.

Qing Lyu, Menglei Chai, Xiang Chen, and Kun Zhou. 2022. Real-Time Hair Simulation With Neural Interpolation. *IEEE Trans. Vis. Comput. Graph.* 28, 4 (2022), 1894–1905.

Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. 2021. Modulated Periodic Activations for Generalizable Local Functional Representations. In *ICCV 2021*. 14194–14203.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV 2020*, Vol. 12346. 405–421.

Giljoo Nam, Chenglei Wu, Min H. Kim, and Yaser Sheikh. 2019. Strand-Accurate Multi-View Hair Capture. In *CVPR 2019*. 155–164.

Kyle Olszewski, Duygu Ceylan, Jun Xing, Jose Echevarria, Zhili Chen, Weikai Chen, and Hao Li. 2020. Intuitive, Interactive Beard and Hair Synthesis With Generative Models. In *CVPR 2020*. 7444–7454.

Sylvain Paris, Héctor M. Briceño, and François X. Sillion. 2004. Capture of hair geometry from multiple images. *ACM Trans. Graph.* 23, 3 (2004), 712–719.

Sylvain Paris, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. 2008. Hair photobooth: geometric and photometric acquisition of real hairstyles. *ACM Trans. Graph.* 27, 3 (2008), 30.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *ICLR 2016*.

Qiaomu Ren, Haikun Wei, and Yangang Wang. 2021. Hair Salon: A Geometric Example-Based Method to Generate 3D Hair Data. In *ICIG 2021*, Vol. 12890. 533–544.

Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. 2022. Neural Strands: Learning Hair Geometry and Appearance from Multi-view Images. In *ECCV 2022*, Vol. 13693. 73–89.

Shunsuke Saito, Liwen Hu, Chongyang Ma, Hikaru Ibayashi, Linjie Luo, and Hao Li. 2018. 3D hair synthesis using volumetric variational autoencoders. *ACM Trans. Graph.* 37, 6 (2018), 208.

Yuefan Shen, Changgeng Zhang, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2021. DeepSketchHair: Deep Sketch-Based 3D Hair Modeling. *IEEE Trans. Vis. Comput. Graph.* 27, 7 (2021), 3250–3263.

Tiancheng Sun, Giljoo Nam, Carlos Aliaga, Christophe Hery, and Ravi Ramamoorthi. 2021. Human Hair Inverse Rendering using Multi-View Photometric data. In *EGSR 2021*. 179–190.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR* abs/1607.08022 (2016).

Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. 2009. Example-based hair geometry synthesis. *ACM Trans. Graph.* 28, 3 (2009), 56.

Ziyan Wang, Giljoo Nam, Tuur Stuyck, Stephen Lombardi, Michael Zollhöfer, Jessica K. Hodgins, and Christoph Lassner. 2022. HVH: Learning a Hybrid Neural Volumetric Representation for Dynamic Hair Performance Capture. In *CVPR 2022*. 6133–6144.

Yichen Wei, Eyal Ofek, Long Quan, and Heung-Yeung Shum. 2005. Modeling hair from multiple views. *ACM Trans. Graph.* 24, 3 (2005), 816–820.

Sebastian Winberg, Gaspard Zoss, Prashanth Chandran, Paulo F. U. Gotardo, and Derek Bradley. 2022. Facial hair tracking for high fidelity performance capture. *ACM Trans. Graph.* 41, 4 (2022), 165:1–165:12.

Keyu Wu, Yifan Ye, Lingchen Yang, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2022. NeuralHDHair: Automatic High-fidelity Hair Modeling from a Single Image Using Implicit Neural Representations. In *CVPR 2022*. 1516–1525.

Zexiang Xu, Hsiang-Tao Wu, Lvdi Wang, Changxi Zheng, Xin Tong, and Yue Qi. 2014. Dynamic hair capture using spacetime optimization. *ACM Trans. Graph.* 33, 6 (2014), 224:1–224:11.

Lingchen Yang, Zefeng Shi, Youyi Zheng, and Kun Zhou. 2019. Dynamic hair modeling from monocular videos using deep neural networks. *ACM Trans. Graph.* 38, 6 (2019), 235:1–235:12.

Meng Zhang, Menglei Chai, Hongzhi Wu, Hao Yang, and Kun Zhou. 2017. A data-driven approach to four-view image-based hair modeling. *ACM Trans. Graph.* 36, 4 (2017), 156:1–156:11.

Qing Zhang, Jing Tong, Huamin Wang, Zhigeng Pan, and Ruigang Yang. 2012. Simulation Guided Hair Dynamics Modeling from Video. *Comput. Graph. Forum* 31, 7 (2012), 2003–2010.

Yujian Zheng, Zirong Jin, Moran Li, Haibin Huang, Chongyang Ma, Shuguang Cui, and Xiaoguang Han. 2023. HairStep: Transfer Synthetic to Real Using Strand and Depth Maps for Single-View 3D Hair Modeling. In *CVPR 2023*. 12726–12735.

Yi Zhou, Liwen Hu, Jun Xing, Weikai Chen, Han-Wei Kung, Xin Tong, and Hao Li. 2018. HairNet: Single-View Hair Reconstruction Using Convolutional Neural Networks. In *ECCV 2018*, Vol. 11215. 249–265.

A ADDITIONAL RESULTS

In Fig. 17, we present the raw outputs of the neural upsampler to evaluate its densification power as an individual module. In Fig. 18 we demonstrate various combinations of the user parameters.

B STRAND INTERPOLATION

In Fig. 19, we compare interpolation trajectories of two distinct strands in different latent spaces. Interpolation in the spatial latent space demonstrates unstable curvature changes, while interpolation in the frequency latent space aligns better with human perception.

Moreover, we quantitatively examine the jittering effect during strand interpolation with different representations. We randomly select 10K strand paris from the test set and perform interpolation between them with 98 intermediate sample points. For each interpolation sequence, we then compute the *jitter* metric [Flash and Hogan 1985], defined as the third derivative of position by assuming the time interval is 1 second, where a smaller value means a smoother transition. As shown in Tab. 6, the transition in the frequency latent representation is as smooth as the spatial-latent representation with better curvature preservation. This suggests that the frequency-latent space is better structured.

C PENETRATION REFINEMENT

Our system is based on the same head geometry and the detailed head shape is not explicitly considered. Although the shape information is partially baked into the weights of the neural networks, penetrations still happen occasionally. To mitigate the penetration artifacts and preserve the hair structure as much as possible, the following refinement is performed. For each strand in the Euclidean space, we traverse its vertices from root to tail. At vertex $\boldsymbol{p}_i$, we check if the vertex ahead, $\boldsymbol{p}_{i+k}$, is within the head mesh using a pre-computed signed distance field. If $\boldsymbol{p}_{i+k}$ is inside the mesh, we compute the minimal rotation angle θ that pushes $\boldsymbol{p}_{i+k}$ out of the mesh along the normal $\boldsymbol{n}$ of the nearest surface with $\boldsymbol{p}_i$ as the rotation pivot and $\boldsymbol{b} = (\boldsymbol{p}_{i+k} - \boldsymbol{p}_i) \times \boldsymbol{m}$ as the axis. The vertices $\boldsymbol{p}_j (j >= i)$ are rotated by $\theta_i = \theta * \delta^{j-i-k/2}$. After traversing all the vertices, we remove the strands that still penetrate the head mesh. We empirically set $k = 20$, $\delta = 0.9$ for all experiments.

Fig. 17. Raw outputs from our neural upsampler. It effectively populates the guide hairs, preserves the shape, keeps the parting lines, and avoids unnatural patterns.

Table 6. The *jitter* metrics of strand interpolations in different domains. While vanilla Euclidean interpolation is most smooth, our latent representation in the frequency space is similarly smooth as the spatial one.

	Euclidean	spatial-latent	frequency-latent
jitter (mm/s^3)	0.69	0.89	0.97

Table 7. Detail structure of the hairstyle-VAE model (encoder part). The decoder is symmetric. The residual connections are between layers 1 & 3, 4 & 6, and 7 & 9 using bilinear downsampling. Layer 11 gives the final output with 1024 channels, half of which represents the latent vector while the other half is the log variation used for the reparameterization trick in VAE training. Layer 12 is used in the residual connection between layer 1 and 3 to align the number of channels after downsampling.

layer number	input size	convolution
1	$24 \times 32 \times 65$	(1, 1, 65, 2048, 1)
2	$24 \times 32 \times 2048$	(3, 3, 2048, 2048, 2)
3	$12 \times 16 \times 2048$	(1, 1, 2048, 512, 1)
4	$12 \times 16 \times 512$	(1, 1, 512, 2048, 1)
5	$12 \times 16 \times 2048$	(3, 3, 2048, 2048, 2)
6	$6 \times 8 \times 2048$	(1, 1, 2048, 512, 1)
7	$6 \times 8 \times 512$	(1, 1, 512, 2048, 1)
8	$6 \times 8 \times 2048$	(3, 3, 2048, 2048, 2)
9	$3 \times 4 \times 2048$	(1, 1, 2048, 512, 1)
10	$3 \times 4 \times 512$	(3, 4, 512, 1024, 1)
11	$1 \times 1 \times 1024$	(1, 1, 1024, 1024, 1)
12	$12 \times 16 \times 65$	(1, 1, 65, 512, 1)

Table 8. Detail structure of the neural upsampler (generator part). The discriminator is the same except the first and last layers. The residual connections are between layers 4 & 6, 7 & 9, and 10 & 12.

layer number	input size	convolution
1	$216 \times 288 \times 364$	(1, 1, 128, 364, 1)
2	$216 \times 288 \times 128$	(13, 13, 128, 128, 1)
3	$216 \times 288 \times 128$	(1, 1, 128, 128, 1)
4	$216 \times 288 \times 128$	(1, 1, 128, 128, 1)
5 - 6	same as layer 2 - 3	
7 - 9	same as layer 4 - 6	
10 - 11	same as layer 4 - 5	
12	$216 \times 288 \times 128$	(1, 1, 128, 5, 1)

D NETWORK STRUCTURES

In Tab. 7 and Tab. 8 we provide the detailed structure of our hairstyle-VAE and neural upsampler, respectively. The input size is formatted as $h \times w \times c$ where h, w, and c are height, width, and channels. The convolution is formatted as (kernel height, kernel width, input channels, output channels, stride).

E QUALITATIVE COMPARISON WITH VHV

To qualitatively compare our model with the seminal work of VHV [Saito et al. 2018], in Fig. 20, we present VHV reconstruction results of the rendered hair models generated by our method. While

Fig. 18. Hair models from the same guide strands but varied user parameters.

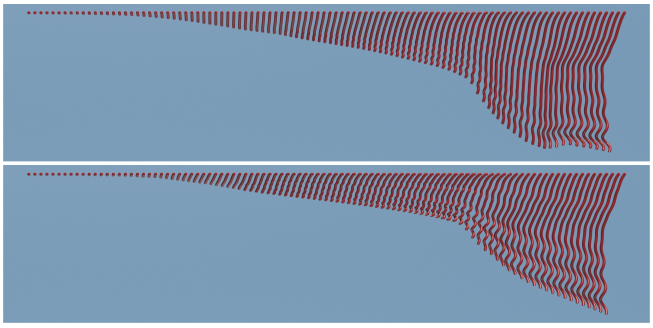

Fig. 19. Interpolation trajectories between a short strand and a long wavy strand in different latent spaces. Top: spatial latent space; Bottom: frequency latent space. The frequency representation preserves the curvature better.

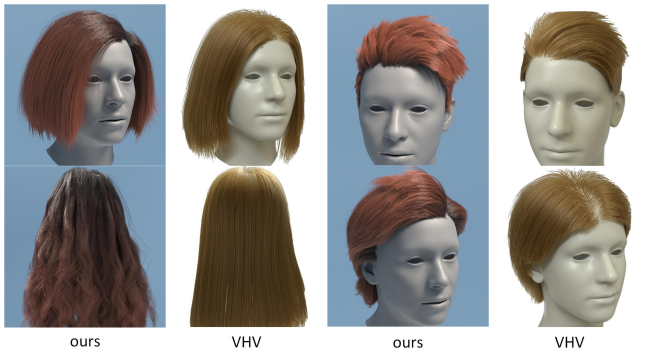

Fig. 20. VHV reconstruction results of the rendered hair models generated by our method. While the overall hairstyles are successfully recovered, the reconstructions suffer from over-smoothing and lack of high-fidelity details.

VHV can recover the overall hairstyle well, some strand- and wisp-level details are missing, partially due to the limited capability of volumetric representations. Furthermore, some styles, such as the bottom-right hair model, are beyond the coverage of VHV. Note that this is not a strictly fair comparison as VHV is reconstructing the hair model only from a single-view image, and the face alignment step in the original method does not work on our non-photorealistic face rendering.

F DATASET

In Fig. 21 we visualize the complete list of 35 base hairstyle categories that constitute our dataset, which covers a wide variety of hairstyles.

Fig. 21. All 35 base hairstyle categories of our dataset.