Neural Strands: Learning Hair Geometry and Appearance from Multi-View Images

Radu Alexandru Rosu[1⋆], Shunsuke Saito[3], Ziyan Wang[2,3], Chenglei Wu[3], Sven Behnke[1], and Giljoo Nam[3]

[1] University of Bonn, Germany
[2] Carnegie Mellon University
[3] Reality Labs Research

radualexandru.github.io/neural_strands

Abstract. We present Neural Strands, a novel learning framework for modeling accurate hair geometry and appearance from multi-view image inputs. The learned hair model can be rendered in real-time from any viewpoint with high-fidelity view-dependent effects. Our model achieves intuitive shape and style control unlike volumetric counterparts. To enable these properties, we propose a novel hair representation based on a neural scalp texture that encodes the geometry and appearance of individual strands at each texel location. Furthermore, we introduce a novel neural rendering framework based on rasterization of the learned hair strands. Our neural rendering is strand-accurate and anti-aliased, making the rendering view-consistent and photorealistic. Combining appearance with a multi-view geometric prior, we enable, for the first time, the joint learning of appearance and explicit hair geometry from a multi-view setup. We demonstrate the efficacy of our approach in terms of fidelity and efficiency for various hairstyles.

1 Introduction

Photorealistic rendering of digital humans plays an important role in many AR/VR applications such as virtual telepresence. In recent years, data-driven approaches have shown compelling results on geometry and appearance modeling of digital humans, especially for face [21, 19, 41, 43] and body [36, 2, 46]. Hair, on the other hand, still remains a challenge due to the sheer number of thin hair strands, their complex geometric structures, and non-trivial light transport effects such as subsurface scattering and specular reflections at microscale.

To enable strand-accurate hair reconstruction, a recent work [28] leveraged explicit line assumption in multi-view stereo reconstruction. However, the reconstruction does not provide complete hair strands from the root on the scalp due to heavy self-occlusions. To date, connecting line segments from the scalp to the tip of hair for a variety of hairstyles remains difficult without strong data prior.

⋆ Work done during an internship at Reality Labs Research, Pittsburgh, PA, USA.

Fig. 1: Given multi-view images we recover both explicit geometry (left) and photo-realistic appearance of hair that generalizes to novel views (middle).

Appearance modeling of hair is also an active research field [17, 4]. Physics-based rendering approaches typically require extensive light-transport computation to represent complex appearance of 3D hair strands, hence are too slow for real-time applications. Recently, data-driven approaches [6, 45] enable photorealistic rendering from geometric proxies such as orientation fields using neural rendering techniques. However, due to sub-optimal geometric quality and feature representations, these image-space neural rendering methods typically suffer from view-inconsistency and lack of fidelity. Volumetric rendering techniques [27, 23], on the other hand, achieve view-consistent novel-view rendering, but geometry-driven manipulation is not possible.

In this work, we present Neural Strands, a novel learning framework for jointly modeling hair geometry and appearance, which can be readily used for real-time rendering of photorealistic hair from an arbitrary viewpoint. Our idea is to build a strong data prior using a strand-based generative model learned from synthetic data. This allows us to register complete hair strands from the partial hair reconstruction obtained by [28]. To parameterize the appearance and geometry of complete hairstyles from registration, we further present a novel hair representation called neural scalp textures, where each texel on a UV texture stores a feature vector describing both the shape and appearance of a single strand at a corresponding scalp position. With the aforementioned strand generator, the neural scalp texture is decoded into dense 3D strands, which are then rendered into RGB images by a neural renderer.

Our strand generator is a multilayer perceptron network that takes as input a strand shape feature vector and outputs the 3D shape of the strand. Inspired by neural ordinary differential equations [10], we design our strand generator to yield the first-order derivatives of a strand geometry, i.e., directional vectors with magnitudes. A complete 3D strand shape can be obtained by integrating the derivatives. This formulation allows for generation of smooth strands and also enables a trade-off between compute and accuracy by changing the integration step size. To learn a generic strand generator, we pre-train the network on a

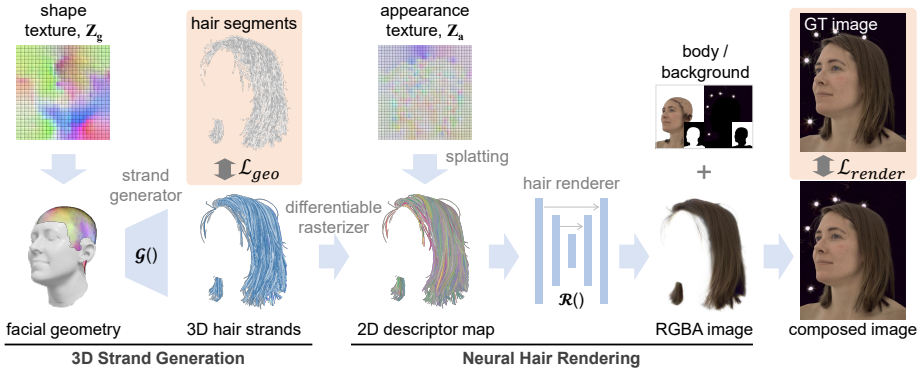


Fig. 2: System overview. A neural scalp texture describing the strand shapes is embedded onto the scalp. A strand generator decodes the shape descriptors into explicit strands. The 3D strands are then rasterized to the screen space. Finally, a hair renderer decodes the 2D descriptors into an RGBA image of the hair which is composited with the body and background. We train our system end-to-end with both a geometrical loss towards sparse line segments and a rendering loss towards the ground-truth images.

wide variety of strand data, resulting in a generic strand prior for robust strand fitting to the noisy real-world scan data.

In order to render photo-realistic hair images using the generated strands, we propose a neural hair renderer. The renderer comprises two parts: a differentiable rasterizer and an image synthesizer. The rasterizer splats the appearance feature vectors onto 2D images while being differentiable w.r.t. the 3D positions of strands and the splatted features. The image synthesizer is a UNet architecture that takes as input the rasterized feature map and yields the final rendered images. We train all the components in an end-to-end manner with direct image supervision. With an explicit strand representation, our method can directly edit or control the hair for rendering, e.g. for virtual haircut or hair blowing, which differentiates our approach from the recent work on free-viewpoint volumetric rendering [22, 23, 27, 31].

In summary, our main contributions are:

- the first joint learning framework for high-quality strand geometry and appearance from multi-view images, which can be readily used for photo-realistic real-time rendering of human hair,
- a highly expressive strand generative model, which enables strand-level hair registration from partial scan data,
- a novel neural scalp texture representation that compactly models explicit hair strand geometry and view-dependent appearance for real-time rendering
- an experiment showing that our approach enables intuitive manipulation of 3D hair and its rendering, which remains challenging for volumetric approaches.

2 Related Work

2.1 Image-based Hair Modeling

Strand-based Representation. A common geometric representation for hair modeling is a set of 3D strands, where a strand is parameterized by a sequence of connected 3D points. To obtain 3D hair strands, typically multi-view hair capture methods first reconstruct a low-resolution base geometry using various image-based 3D modeling techniques, and run an additional strand-fitting process to obtain dense 3D hair strands that are connected to the scalp [3, 24, 25, 13, 14, 12, 11]. However, these methods often lack fine details like flyaway hairs because of the low-resolution base geometry. To overcome this limitation, Nam *et al.* [28] proposed a line-based multi-view stereo (LMVS) that is tailored to hair capture tasks and directly obtains 3D line segments from images. Sun *et al.* [38] further improved LMVS and estimated the reflectance parameters of hair for photo-realistic rendering. However, the reconstructed strands from both methods [28, 38] only cover the outer surface of hair and they are not connected to the scalp. Another line of research focuses on single-view hair modeling [9, 8, 5, 7, 49, 52, 50, 47]. While these works show promising results from less constrained capture setups, due to the ill-posed nature, these methods do not provide metrically accurate 3D geometry of hair strands. Moreover, they are not directly applicable to novel-view synthesis in a photorealistic manner. In contrast, our approach jointly learns metrically accurate geometry and appearance from multi-view images, enabling rendering from any viewpoint.

Volumetric Representation. Volumetric representation is also used for hair modeling. Saito *et al.* [35] propose to regress 3D hair from a single image using volumetric orientation fields as an intermediate representation, which can be easily handled by 3D convolutions. However, due to the low grid resolution of voxels, fine details such as flyaway hairs are not well represented. Combined with differentiable volumetric rendering, Neural Volumes [22] and NeRF [27] enable highly expressive geometry and appearance modeling of objects from multi-view images. Mixture of volumetric primitives (MVP) addresses the performance bottleneck of volumetric representations by representing scenes as a collection of small voxels [23]. While these methods enable realistic rendering of 3D hairs, lack of geometric hair control hinders us from driving and intuitive manipulation of photorealistic hair models.

2.2 Neural Hair Rendering

Neural rendering [40] has recently gained great attention for rendering photorealistic images. Given 2D segmentation masks [39] and 2D orientation maps [30, 15, 33, 39], generative adversarial networks (GANs) are trained to create RGB hair images that match the input data. By rendering these 2D features from 3D hair strands, these approaches allow us to photorealistically render hair images as well [45, 6]. However, we observe that rendering quality of these image-based approaches is highly dependent on the conditioned 2D features, and often leads to

view-inconsistent results with limited fidelity. In this work, we show that highly accurate strands with a per-strand appearance code improve view consistency and fidelity of neural hair rendering.

3 Overview

Fig. 2 shows the overview of our learning framework. Neural Strands consists of three parts: neural scalp textures, strand generator, and neural hair renderer. A neural scalp texture is a 2D UV-texture that stores either the shape (shape texture, $\mathbf{Z}_g$) or the appearance (appearance texture, $\mathbf{Z}_a$) of strands. The strand generator, $\mathcal{G}()$, is a generative neural network that transforms a strand feature vector into a 3D strand geometry. Finally, the hair renderer, $\mathcal{R}()$, is a UNet-architecture that renders hair images from rasterized appearance feature maps.

The design of Neural Strands is motivated by several hair-specific attributes. First, in terms of geometry, each strand has its own attachment point in 3D scalp position. By using an explicit scalp UV-map, we can easily control or edit the geometric features of strands based on the scalp location. The shape texture $\mathbf{Z}_g$ exploits this property and enables several applications such as virtual haircut and hairstyle manipulation. Second, while hair shows extremely complex appearance in 2D images, individual strands have a smooth color variation along their strand directions. The complex hair appearance is determined by how the strands are shaped in 3D and how they are projected to each viewpoint. Therefore, we only store the low-frequency appearance information of each strand using our appearance texture $\mathbf{Z}_a$. The high-frequency appearance in 2D images can be effectively represented by our rasterizer and hair renderer. This separation of per-strand appearance modeling and rendering is one of the keys that enable our high-fidelity results. In the following sections, we explain the details of each component and how they form the final hair images (Section 4) and then present the training procedure of our learning framework (Section 5).

4 Neural Strands

4.1 Neural Scalp Textures

Neural scalp textures are our base hair representation for explicit 3D geometry and appearance of strands. Each texel in the shape texture $\mathbf{Z}_g$ stores a feature vector $\mathbf{z}_g^i \in \mathbb{R}^{D_g}$ that conveys the shape information of a single hair strand. Since the strand roots are created in 3D to ensure uniform coverage, the strands sample their corresponding $\mathbf{z}_g$ from the scalp texture using bilinear interpolation. Similarly, the appearance texture is denoted as $\mathbf{Z}_a$ and $\mathbf{z}_a^i \in \mathbb{R}^{D_a}$ stores the appearance information of the strand. For the remaining of this paper, we assume that the scalp UV-mapping is known in advance. We set the texture resolutions to 256^2 for $\mathbf{Z}_g$, 512^2 for $\mathbf{Z}_a$, and $D_g = 64$, $D_a = 16$.

4.2 Strand Generator

A single hair strand $\mathbf{S}$ is a sequence of 3D points: $\mathbf{S} = \{\mathbf{p}_k\}_{k=0}^{L}$, where L is set to 100. The full hair shape is a collection of strands $\{\mathbf{S}^j\}$, where j indexes over the strands. Our strand generator $\mathcal{G}()$ is a neural network that transforms a shape feature vector $\mathbf{z}_g$ into a full 3D strand geometry $\mathbf{S}$, i.e., $\mathcal{G}() : \mathbb{R}^{D_g} \rightarrow \mathbb{R}^{3\times L}$.

Network Architecture. Inspired by NeuralODEs [10], we model the strand generation as an integration of hair gradients along each strand. We thus design the network to output finite differences along hair growing directions, i.e., $\mathbf{d}_k = \mathbf{p}_k - \mathbf{p}_{k+1}$, instead of 3D positions $\mathbf{p}_k$.

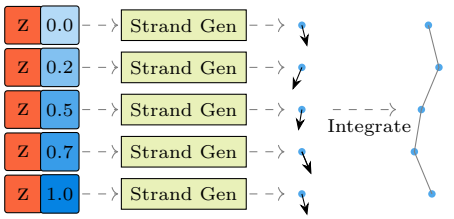


Fig. 3: The strand generator takes as input the shape descriptor z and a parameter $t \in [0, 1]$ indicating the position between root and tip. It outputs a direction vector for each section of the strand. The directions are integrated from root to tip in order to recover the explicit 3D positions of the strand nodes.

In order to model the high-frequency geometric details, we implement our strand generator using the modulated SIREN [26]. Our strand generator has two multilayer perceptron (MLP) networks: a modulator and a synthesizer. The modulator takes as input the strand shape feature $\mathbf{z}_g$ and outputs modulation vectors that modify the amplitude, frequency, and phase shift of the sinusoidal activation functions of the synthesizer network. The SIREN[37]-based synthesizer takes as input a parameter $t \in [0, 1]$ that indicates the relative position along the strand from root (0) to tip (1). The output of the synthesizer is a 3D directional vector $\mathbf{d}_t$ with magnitude. The final positions of the strand nodes are obtained by the forward Euler method, i.e., $\mathbf{p}_k = \sum_{i=0}^{k} \mathbf{d}_i$. See Fig. 3 for illustration.

Discussion. The benefit of working in the gradient domain is that each node of the strand has only local effect. When working directly with the positions, rotating the hair along the root node modifies the positions of all the subsequent nodes of the strand since they can all be considered as a kinematic chain. However, in the gradient domain, modifying the root node direction $\mathbf{d}_0$ does not necessarily modify the rest of the directions. This independence between nodes makes the learning easier, as the network does not need to learn the kinematic dependency between hair nodes. In addition, long straight hair can be easily represented by a network which outputs mostly constant directions.

4.3 Neural Hair Renderer

Since we have an explicit hair geometry $\{\mathbf{S}^j\}$ from our strand generator, we now render the hair appearance using a differentiable rasterization-based neural renderer. We take inspiration from the Neural Point-Based Graphics (NPBG) [1] and the Deferred Neural Rendering (DNR) [42] which use a point/mesh rasterizer. Both NPBG and DNR use a geometry proxy (points or mesh) in order to

carry a neural descriptor either with a texture map or at a per-point level. The neural descriptors are rasterized to a screen space and a UNet regresses the final color image.

In our case, the geometry proxy is a 3D line segment, and the neural descriptors are given by the appearance texture $\mathbf{Z}_a$. However, hair has various properties that need to be properly addressed. First, hair is not opaque and can show complex scattering effects. Second, the thin geometry also tends to create aliasing artifacts when rendered. We effectively solve these issues by using alpha blending and let the UNet renderer directly output the alpha maps for natural strand color blending.

Neural Hair Descriptor. The appearance texture $\mathbf{Z}_a$ stores the vectors $\mathbf{z}_a^i$ for each texel position i. For each strand $\mathbf{S}^j$, we bi-linearly interpolate the neighbouring four texels at the root to obtain the corresponding $\mathbf{z}_a$ for the strand. The 3D points $\{\mathbf{p}_k\}_{k=0}^{L}$ belonging to the same strand $\mathbf{S}^j$ share the same per-strand feature vector $\mathbf{z}_a$. A per-point neural descriptor is then defined as a concatenated vector of the strand feature, the point direction $\mathbf{d}$, and the t parameter:

$$\mathbf{g} = [\mathbf{z}_a, \mathbf{d}, t], \quad \mathbf{g} \in \mathbb{R}^{D_a+3+1}. \tag{1}$$

The per-point descriptors are rasterized to each viewpoint via a differentiable line rasterizer.

Differentiable Rasterization. We project the neural descriptors to screen-space by rasterizing the strand lines. However, as naïve rasterization methods are non-differentiable, we replace the hard rasterizer with soft-rasterization [34, 20, 48]. We first hard-rasterize unique strand indices onto the screen using OpenGL which allow to recover for each pixel a 3D point that lies on a non-occluded strand line. The 3D point is associated with a descriptor $\tilde{\mathbf{g}}$ by linearly interpolating the descriptors from the two points that define the strand segment. At the end of this step, we obtain a point cloud of the visible points of the hair together with their neural descriptors. As a second step, we project the cloud to screen and the descriptors are bi-linearly splatted to the neighboring pixels. In the case where multiple 3D points contribute to the same pixel, the splatted descriptor at the pixel is defined as the weighted average of the contributing 3D points:

$$\mathbf{h}_u = \frac{\sum w \cdot \tilde{\mathbf{g}}}{\sum w}, \tag{2}$$

where the subscript u is the pixel index in the rasterized image space, $\mathbf{h}_u$ is the rasterized and averaged descriptor at the pixel, and w is the contribution weight of each descriptor $\tilde{\mathbf{g}}$. This soft-rasterization for hair is crucial to ensure the rendering loss to be back-propagated to the neural scalp textures $\mathbf{Z}_g$ and $\mathbf{Z}_a$ as in Fig. 2. In order to deal with possible holes in the hair, we also splat at multiple resolutions and concatenate each resolution with the corresponding layer in the UNet, similar to the previous works [1, 34].

Image Generation. The multi-resolution descriptor maps are concatenated with per-pixel viewing directions in order to model view-dependent effects and are given as input to the UNet which predicts an RGB and an opacity map for

the hair. Effectively, the input to the UNet is a descriptor map of $(D_a + 3 + 1 + 3)$ channels concatenated to each downsampling stage of the network, and the output is a four-channel image of RGB and alpha.

We find that the intermediate activation maps of the UNet were aliased by the down-sampling with strided convolutions. Therefore, we replace the down-sampling and up-sampling layers of the UNet with the anti-aliased versions used in [16] and the activation function with their filtered leaky ReLU. We find this change effectively solves the aliasing issue and removes temporal flickering artifacts when rendering novel-view images.

Image Composition. In order to blend the hair with the background and body parts, we also learn a low-resolution texture for a body mesh and a background mesh represented as a sphere around the subject. The background, body, and hair are alpha-blended together in order to recover the full image. The compositing can be viewed in Fig. 2.

5 Training

Training Neural Strands is twofold. First, we pre-train the strand generator $\mathcal{G}()$ using synthetic hair models and freeze the parameters. Then, for each subject, we optimize the feature vectors of neural scalp textures $\mathbf{Z}_g$ and $\mathbf{Z}_a$, as well as the parameters of the UNet renderer $\mathcal{R}()$ using the pre-trained generator $\mathcal{G}()$.

5.1 Strand Generator

VAE Training. The role of our strand generator $\mathcal{G}()$ is to provide a strong prior of realistic hair strand shapes that can be readily used for our image-based hair modeling framework. To this end, we train it in an auto-encoder fashion with synthetic 3D curves. Concretely, we implement it as a variational autoencoder (VAE) [18] in order to obtain a smooth embedding of $\mathbf{z}_g$. The input and output of the VAE is a strand, i.e., $\{\mathbf{p}_k\}_{k=0}^{L}$.

We design a simple encoder network with a 1D CNN. Given the 3D points of a strand, the encoder outputs the parameters $\mathbf{s}_\mu$ and $\mathbf{s}_\sigma$ of the Gaussian distribution over the latent variables. During training, we sample $\mathbf{z}_g$ from this distribution using the reparameterization trick: $\mathbf{z}_g = \mathbf{s}_\mu + \epsilon \cdot \mathbf{s}_\sigma, \epsilon \sim \mathcal{N}(0, 1)$. Given the strand embedding $\mathbf{z}_g$, we decode it back to the original points using the decoder $\mathcal{G}()$ which is implemented as a modulated SIREN.

We use the L2 loss between the predicted and ground-truth 3D points. Since this loss gives little regard to high-frequency detail like curls, we add a loss on the predicted directions. The data term is defined as

$$\mathcal{L}_{data} = \sum_{i=0}^{L} \|\mathbf{p}_i - \tilde{\mathbf{p}}_i\|_2^2 + \lambda_d \left(1 - \mathbf{d}_i \cdot \tilde{\mathbf{d}}_i\right), \tag{3}$$

where $\mathbf{p}$ and $\tilde{\mathbf{p}}$ are the original and reconstructed points, and $\mathbf{d}$ and $\tilde{\mathbf{d}}$ are their directions. λ_d is set to 1×10^{-3}. We also add the Kullback–Leibler divergence

term $\mathcal{L}_{KL}$ [18]. We train the VAE with the total loss:

$$\mathcal{L}_{VAE} = \mathcal{L}_{data} + \lambda_{KL}\mathcal{L}_{KL}\left(\mathcal{N}(\mathbf{s}_\mu, \mathbf{s}_\sigma) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I})\right), \tag{4}$$

where λ_{KL} is set to 1×10^{-3}. Once the VAE is trained, we discard the encoder network and only use the decoder as our pre-trained strand generator $\mathcal{G}()$.

Training Data. The dataset to train the strand generator is a set of synthetic 3D curves and each curve represents a hair strand as a sequence of 100 points. To remove the variance between the strands, we represent each one in a local coordinate system defined by the root position and the tangent-bitangent-normal (TBN) at the scalp. We also augment each strand by randomly stretching each dimension, mirroring along the tangent and bitangent vectors and rotating along the normal.

5.2 End-to-End Optimization

Data Preparation. Given multi-view images as input, we first perform multi-view stereo to obtain 3D geometry of the subject. We then fit the reconstructed face geometry to the FLAME face template [19]. This fitting process gives us a known UV-mapping for the scalp region and also effectively removes the hair geometry in the reconstructed mesh. We also perform the line-based multi-view stereo (LVMS) [28] to get partial hair strand reconstruction. Note that the partial strands only include the strands in the outer surface of hair and are not connected to the scalp. We additionally perform a diffusion algorithm based on user strokes similar to [8] in order to resolve the directional ambiguity of the line segments and obtain a consistent direction of growth.

End-to-end training. The input to our end-to-end optimization framework are 1) multi-view images, 2) the fitted facial geometry, and 3) the partial hair strands. Given the pre-trained strand generator $\mathcal{G}$ and input data, we jointly optimize for the neural scalp textures $\mathbf{Z}_g$ and $\mathbf{Z}_a$ as well as the parameters of the UNet renderer $\mathcal{R}()$ for each captured subject.

Geometric Loss. The geometric loss encourages our strand generator $\mathcal{G}()$ to output hair strands that align with the partial hair strands from the LMVS [28]. The loss is defined as the bi-directional Chamfer of the distance and directions between the two point clouds:

$$\mathcal{L}_{geo} = \sum_{\mathbf{x}\in X}\Big(\|\mathbf{x} - \mathbf{y}_\mathbf{x}\|_2 + (1 - \mathbf{d}_x \cdot \mathbf{d}_y)\Big) + \sum_{\mathbf{y}\in Y}\Big(\|\mathbf{x}_\mathbf{y} - \mathbf{y}\|_2 + (1 - \mathbf{d}_x \cdot \mathbf{d}_y)\Big), \tag{5}$$

where X is a set of 3D points in the generated strands, Y is a set of points in the LMVS-reconstructed hair segments, and $\mathbf{y}_\mathbf{x}$ represents the point $\mathbf{y}$ which is the closest one in Y to the point $\mathbf{x}$. Effectively, the Chamfer distance brings the closest points closer together and also aligns their directions.

While the geometric loss alone gives us plausible hair geometry, it can lead to missing or sub-optimal fitting results as the LMVS-reconstructed hair segments do not cover the entire region of the hair. Because the strands that are not

reconstructed from the LMVS could still be visible from the images, there is an opportunity to supervise the strand generation with the rendering loss. We therefore also use the rendering loss to optimize for the shape texture $\mathbf{Z}_g$. This is done by back-propagating the rendering loss not only to the appearance texture $\mathbf{Z}_a$, but also to the shape texture $\mathbf{Z}_g$.

Rendering Loss. The neural renderer $\mathcal{R}()$ together with the appearance texture $\mathbf{Z}_a$ is trained using a combination of L2 and LPIPS [51] loss. The rendering loss $\mathcal{L}_{render}$ is thus defined as:

$$\mathcal{L}_{render} = \sum_{n=1}^{N} \left(\left\| I_n - \tilde{I}_n \right\|_2^2 + \lambda_L \mathcal{L}_{LPIPS}(I_n, \tilde{I}_n) \right), \tag{6}$$

where I_n and $\tilde{I}_n$ are the rendered and captured images from n-th view, and N is the number of multi-view images. We set $\lambda_L = 0.1$.

Alpha Loss. In order to also offer supervision to the predicted alpha, we rasterize the LMVS line-segments to the image. By itself, this hard LMVS mask would be inadequate to be used as alpha supervision since it enforces a strictly opaque hair. To remedy this, we dilate the mask and define a region without hair that we can claim should be empty and therefore should have an alpha of 0. After the dilation, we erode to define an interior region that we can be certain that it should be opaque. These two regions are used for supervision, while the border regions containing stray hairs are left unsupervised as we cannot reliably supervise their soft opacity. The alpha map loss $\mathcal{L}_{alpha}$ is defined as:

$$\mathcal{L}_{alpha} = \sum_{n=1}^{N} \left(\left\| A_n - \tilde{A}_n \right\|_2^2 \cdot M_n \right), \tag{7}$$

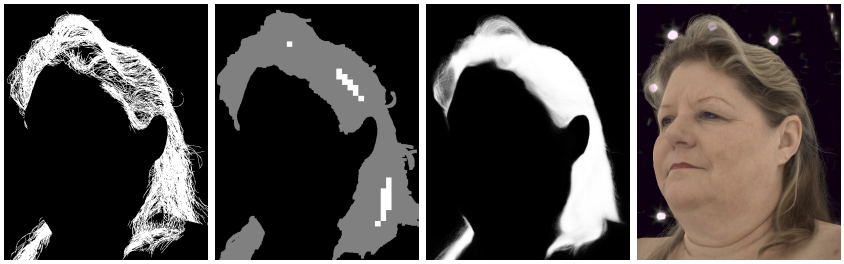
where A_n and $\tilde{A}_n$ are the reference alpha map from LMVS and the generated alpha map from n-th view, and M_n is their mask of union of the interior and exterior regions. See Fig. 4 for illustration.

Total Loss. The total loss for our end-to-end optimization is:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{geo} + \lambda_2 \mathcal{L}_{render} + \lambda_3 \mathcal{L}_{alpha}, \tag{8}$$

where λ_1,λ_2, and λ_3 are set to 1, 1×10^{-3}, and 1×10^{-3}, respectively.

Optimization Details. Fitting the geometry of the strands based on the Chamfer loss to the line-segments is a highly ill-posed problem. To solve this, we propose two solutions: a coarse-to-fine and a root-to-tip optimization of the shape texture $\mathbf{Z}_g$. Coarse-to-fine optimization of the scalp imposes a smoothness prior on the features. We implement this by sampling from the texture map in the forward pass and blurring the gradient of the loss w.r.t to the texture $\frac{\nabla \mathcal{L}}{\nabla \mathbf{Z}_g}$ in the backward pass, so that neighboring pixels receive similar gradients. We start by blurring with a large kernel and gradually decrease it until the gradient is propagated only towards the pixels that correspond to the strand root. This optimization scheme is similar to the Laplacian Pyramid from [42]. However,



(a) LMVS mask (b) Mask trimap (c) Predicted alpha (d) Novel image

Fig. 4: Alpha prediction and blending. (a) Mask from LMVS; (b) Trimap obtained from (a); (c) Predicted alpha map; (d) Composed image using (c).

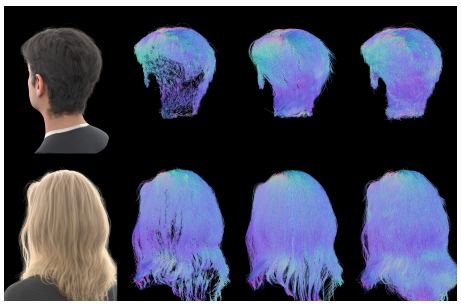
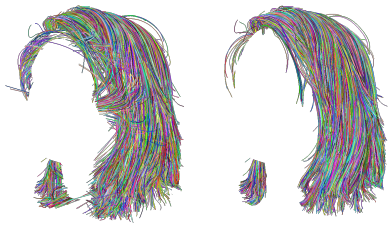


Fig. 5: Synthetic data and geometry reconstructions. From left: GT image, LMVS geometry, our geometry, GT geometry. Note that the strands from LMVS are segmented and not connected to the scalp.



a) No coarse-to-fine nor root-to-tip b) With coarse-to-fine and root-to-tip

Fig. 6: Our coarse-to-fine and root-to-tip optimization of the shape texture helps the network converge to the correct shape.

instead of optimizing various textures at multiple resolutions, we optimize only one, which makes it faster for training and inference.

Root-to-tip optimization is performed by starting the training with only the roots of the strands and masking out the gradient for the rest of the strand vertices. We linearly anneal the rest of the nodes gradually during optimization. In addition, for stable training, we set λ_2 and λ_3 to 0 for the first 1,000 iterations since at the beginning, the strands are far away from the correct hair region in image-space. In Fig. 6 we show the impact of our coarse-to-fine and root-to-tip optimization scheme.

Table 1: Comparison between the previous work LMVS [28] and our method using synthetic dataset.

	Short hair						Long hair					
τ_p/τ_d	1mm / 10°		2mm / 20°		3mm / 30°		1mm / 10°		2mm / 20°		3mm / 30°	
Method	LMVS	Ours	LMVS	Ours	LMVS	Ours	LMVS	Ours	LMVS	Ours	LMVS	Ours
Precision	**56.91**	52.79	**93.42**	92.94	**98.85**	98.18	26.25	**32.59**	**75.13**	71.40	**93.51**	71.40
Recall	12.11	**13.78**	30.29	**48.38**	46.62	**71.51**	**16.54**	14.62	39.12	**42.06**	54.01	**62.91**
F-score	19.98	**21.85**	45.75	**63.64**	63.36	**82.75**	**20.30**	20.19	51.45	**52.94**	**68.47**	66.89

6 Results

We evaluate our method using both real and synthetic data. For real images, we use a multi-view camera dome with ∼140 cameras uniformly distributed on a sphere of two meter diameter. For synthetic images, we use artist-created 3D hair models. Virtual cameras are placed to mimic the real capture setup. Fig. 5 shows the synthetic renderings of two 3D models with short and long hairstyles. We train our model for 48 h on a single NVIDIA V100 GPU.

6.1 Evaluation with Synthetic Data

Since it is impossible to obtain the ground truth geometry of hair strands from real captured images, we use synthetic data for the evaluation. In Tab. 1, we show quantitative comparison on recovered strand geometries over the state-of-the-art hair geometry reconstruction method [28]. We follow the error metric from [28, 38] and show the precision, recall, and F-scores of the reconstructed 3D point clouds over their ground truth with various threshold values. It is shown that the previous work [28] tends to have better precision (accuracy), whereas our method has better recall (completeness) and F-score values in general. This shows the effectiveness of our method to reconstruct complete hair strands that are connected to the scalp. We further emphasize that LMVS only recovers disjoint strand segments while our method recovers full strands of hair which enables further applications such as animations. Fig. 5 illustrates the limitation of [28] and how Neural Strands overcomes it.

6.2 Evaluation with Real Data

We compare our method with two view-synthesis methods, NeRF [27] and MVP [23], that can model and render hair appearance from captured multi-view images. As shown in Fig. 7, our method is capable of rendering highly detailed hair textures, which is difficult to achieve with the other methods.

Tab. 2 shows quantitative comparisons. We compute PSNR, SSIM [44], and LPIPS [51] for the hair region of nine novel-view images that are not used in training. The numbers show the averaged values of six subjects for each method. While our method shows the best LPIPS loss with better visual quality, PSNR and SSIM values are slightly lower than the other methods. This is also reported in previous works [51, 29, 32] on image quality metric; PSNR and SSIM do not

Fig. 7: We compare our method against NeRF [27] and MVP [23]. We render from novel view not seen during training. Our method can achieve higher hair detail and also recovers fine stray hairs unlike the other approaches.

Table 2: We perform better under the perceptual metric (LPIPS), indicating that ours have more realistic looking hairs.

	NeRF	MVP	Ours
PSNR (↑)	31.71	**32.82**	31.30
SSIM (↑)	0.9383	**0.9599**	0.9452
LPIPS (↓)	0.1598	0.1226	**0.0811**

Table 3: Our method can render a static hair in real-time (>25 fps) and dynamic strands at interactive rates (>13 fps).

	NeRF	MVP	Ours
decode (↓)	-	**20.40**	34.54
render (↓)	-	81.60	**38.82**
total (ms) (↓)	27,910	102.00	**73.36**

properly reflect the perceptual quality of reconstructed images. We also compare the rendering time of each method in Tab. 3. Note that we only need to run the decode step (strand generation) once, and the generated strands can be rendered to novel viewpoints in less than 40 ms, thus achieving real-time rendering of > 25 frames per second. All experiments were conducted on a NVIDIA V100 GPU.

Fig. 8: Hair manipulation. Our explicit strand representation allows to directly manipulate the hair by moving it in various directions (animation) or cutting it to any length (haircut).

6.3 Applications

In this section, we describe two demos that show the ability of post-capture manipulation of hair strands, which differentiates our method from other view-synthesis methods. Please see the supplemental video for better visualization.

Virtual Haircut. By having an explicit strand with the shape texture $\mathbf{Z}_g$, we can trim its length and let the neural renderer infer how the hair would look like at the new length. In Fig. 8, we show that the UNet generalizes and produces realistic appearance even for this hair configuration that was never seen during training.

Animation. The explicit hair strands can also be deformed by slightly modifying the direction between adjacent strand nodes to convey a sense of dynamics of hair blowing in the wind. In Fig. 8, we show examples of animating the hair with the appearance inferred for this novel hair configuration.

Interpretable Strand Generator. As the strands are generated from the latent space of a VAE, we can traverse this latent space to generate novel strands. In the supplemental video, we show that we can traverse each dimension of the latent space and discover interpretable controls for curliness, length, etc.

7 Limitations and Future Work

Here we discuss several exciting future research directions to overcome current limitations of our method. First, for complicated hairstyles like the hair-bun in Fig. 7, it is challenging to infer the exact topology since most of it is occluded. Stronger priors for hairstyles learned from various subjects could help alleviate these issues. Second, although our hair model is fully editable, due to the complicated light transport, the generated appearance may not generalize to large hair movement, since lighting effects, like shadows, may be baked in the learned appearance. We will take it as future work to explore more physics-aware rendering since strand-level geometry is available from our model.

References

1. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural Point-Based Graphics. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16. pp. 696–712. Springer (2020)
2. Bagautdinov, T., Wu, C., Simon, T., Prada, F., Shiratori, T., Wei, S.E., Xu, W., Sheikh, Y., Saragih, J.: Driving-signal aware full-body avatars. ACM Transactions on Graphics (TOG) **40**(4), 1–17 (2021)
3. Beeler, T., Bickel, B., Noris, G., Beardsley, P., Marschner, S., Sumner, R.W., Gross, M.: Coupled 3d reconstruction of sparse facial hair and skin. ACM Transactions on Graphics (ToG) **31**(4), 117 (2012)
4. Benamira, A., Pattanaik, S.: A combined scattering and diffraction model for elliptical hair rendering. In: Computer Graphics Forum. vol. 40, pp. 163–175. Wiley Online Library (2021)
5. Chai, M., Luo, L., Sunkavalli, K., Carr, N., Hadap, S., Zhou, K.: High-quality hair modeling from a single portrait photo. ACM Transactions on Graphics (TOG) **34**(6), 204 (2015)
6. Chai, M., Ren, J., Tulyakov, S.: Neural hair rendering. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16. pp. 371–388. Springer (2020)
7. Chai, M., Shao, T., Wu, H., Weng, Y., Zhou, K.: Autohair: fully automatic hair modeling from a single image. ACM Transactions on Graphics **35**(4) (2016)
8. Chai, M., Wang, L., Weng, Y., Jin, X., Zhou, K.: Dynamic hair manipulation in images and videos. ACM Transactions on Graphics (TOG) **32**(4), 75 (2013)
9. Chai, M., Wang, L., Weng, Y., Yu, Y., Guo, B., Zhou, K.: Single-view hair modeling for portrait manipulation. ACM Transactions on Graphics (TOG) **31**(4), 116 (2012)
10. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. arXiv preprint arXiv:1806.07366 (2018)
11. Herrera, T.L., Zinke, A., Weber, A.: Lighting hair from the inside: A thermal approach to hair reconstruction. ACM Transactions on Graphics (TOG) **31**(6), 146 (2012)
12. Hu, L., Bradley, D., Li, H., Beeler, T.: Simulation-ready hair capture. In: Computer Graphics Forum. vol. 36, pp. 281–294. Wiley Online Library (2017)
13. Hu, L., Ma, C., Luo, L., Li, H.: Robust hair capture using simulated examples. ACM Transactions on Graphics (TOG) **33**(4), 126 (2014)
14. Hu, L., Ma, C., Luo, L., Wei, L.Y., Li, H.: Capturing braided hairstyles. ACM Transactions on Graphics (TOG) **33**(6), 225 (2014)
15. Jo, Y., Park, J.: Sc-fegan: Face editing generative adversarial network with user's sketch and color. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1745–1753 (2019)
16. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. arXiv preprint arXiv:2106.12423 (2021)
17. Khungurn, P., Marschner, S.: Azimuthal scattering from elliptical hair fibers. ACM Transactions on Graphics (TOG) **36**(2), 1–23 (2017)
18. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
19. Li, T., Bolkart, T., Black, M.J., Li, H., Romero, J.: Learning a model of facial shape and expression from 4d scans. ACM Trans. Graph. **36**(6), 194–1 (2017)

20. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7708–7717 (2019)
21. Lombardi, S., Saragih, J., Simon, T., Sheikh, Y.: Deep appearance models for face rendering. ACM Trans. Graph. **37**(4) (Jul 2018). https://doi.org/10.1145/3197517.3201401, https://doi.org/10.1145/3197517.3201401
22. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. ACM Trans. Graph. **38**(4) (Jul 2019). https://doi.org/10.1145/3306346.3323020, https://doi.org/10.1145/3306346.3323020
23. Lombardi, S., Simon, T., Schwartz, G., Zollhoefer, M., Sheikh, Y., Saragih, J.: Mixture of volumetric primitives for efficient neural rendering. ACM Trans. Graph. **40**(4) (Jul 2021). https://doi.org/10.1145/3450626.3459863, https://doi.org/10.1145/3450626.3459863
24. Luo, L., Li, H., Paris, S., Weise, T., Pauly, M., Rusinkiewicz, S.: Multi-view hair capture using orientation fields. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 1490–1497. IEEE (2012)
25. Luo, L., Li, H., Rusinkiewicz, S.: Structure-aware hair capture. ACM Transactions on Graphics (TOG) **32**(4), 76 (2013)
26. Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., Chandraker, M.: Modulated periodic activations for generalizable local functional representations. arXiv preprint arXiv:2104.03960 (2021)
27. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp. 405–421. Springer (2020)
28. Nam, G., Wu, C., Kim, M.H., Sheikh, Y.: Strand-accurate multi-view hair capture. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 155–164 (2019)
29. Nilsson, J., Akenine-Möller, T.: Understanding ssim. CoRR **abs/2006.13846** (2020)
30. Olszewski, K., Ceylan, D., Xing, J., Echevarria, J., Chen, Z., Chen, W., Li, H.: Intuitive, interactive beard and hair synthesis with generative models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7446–7456 (2020)
31. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
32. Patel, Y., Appalaraju, S., Manmatha, R.: Deep perceptual compression. CoRR **abs/1907.08310** (2019), http://arxiv.org/abs/1907.08310
33. Qiu, H., Wang, C., Zhu, H., Zhu, X., Gu, J., Han, X.: Two-phase hair image synthesis by self-enhancing generative model. In: Computer Graphics Forum. vol. 38, pp. 403–412. Wiley Online Library (2019)
34. Rückert, D., Franke, L., Stamminger, M.: Adop: Approximate differentiable one-pixel point rendering. arXiv preprint arXiv:2110.06635 (2021)
35. Saito, S., Hu, L., Ma, C., Ibayashi, H., Luo, L., Li, H.: 3d hair synthesis using volumetric variational autoencoders. ACM Transactions on Graphics (TOG) **37**(6), 1–12 (2018)
36. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In:

Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2304–2314 (2019)
37. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems **33** (2020)
38. Sun, T., Nam, G., Aliaga, C., Hery, C., Ramamoorthi, R.: Human hair inverse rendering using multi-view photometric data (2021)
39. Tan, Z., Chai, M., Chen, D., Liao, J., Chu, Q., Yuan, L., Tulyakov, S., Yu, N.: Michigan: Multi-input-conditioned hair image generation for portrait editing. arXiv preprint arXiv:2010.16417 (2020)
40. Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J.Y., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D.B., Zollhöfer, M.: State of the Art on Neural Rendering. Computer Graphics Forum (EG STAR 2020) (2020)
41. Tewari, A., Zollhofer, M., Kim, H., Garrido, P., Bernard, F., Perez, P., Theobalt, C.: Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 1274–1283 (2017)
42. Thies, J., Zollhöfer, M., Nießner, M.: Deferred neural rendering: Image synthesis using neural textures. ACM Transactions on Graphics (TOG) **38**(4), 1–12 (2019)
43. Tran, L., Liu, X.: Nonlinear 3d face morphable model. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7346–7355 (2018)
44. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Trans. Image Processing **13**(4), 600–612 (Apr 2004), http://dx.doi.org/10.1109/TIP.2003.819861
45. Wei, L., Hu, L., Kim, V., Yumer, E., Li, H.: Real-time hair rendering using sequential adversarial networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 99–116 (2018)
46. Xiang, D., Prada, F., Wu, C., Hodgins, J.: Monoclothcap: Towards temporally coherent clothing capture from monocular rgb video. In: 2020 International Conference on 3D Vision (3DV). pp. 322–332. IEEE (2020)
47. Yang, L., Shi, Z., Zheng, Y., Zhou, K.: Dynamic hair modeling from monocular videos using deep neural networks. ACM Transactions on Graphics (TOG) **38**(6), 1–12 (2019)
48. Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. ACM Transactions on Graphics (TOG) **38**(6), 1–14 (2019)
49. Zhang, M., Chai, M., Wu, H., Yang, H., Zhou, K.: A datadriven approach to four-view image-based hair modeling. ACM Trans. Graph **36**(4), 156 (2017)
50. Zhang, M., Zheng, Y.: Hair-gan: Recovering 3d hair structure from a single image using generative adversarial networks. Visual Informatics **3**(2), 102–112 (2019)
51. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
52. Zhou, Y., Hu, L., Xing, J., Chen, W., Kung, H.W., Tong, X., Li, H.: Hairnet: Single-view hair reconstruction using convolutional neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 235–251 (2018)