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

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

1 Template Mesh Fitting

Using multi-view stereo (MVS), we reconstruct the face and body as a triangular mesh. However, MVS also meshes the hair region which is an undesirable effect for our purpose of generating hair on the scalp. We thus propose to deform the original head mesh so that we obtain a "bald" mesh which can serve as a basis for hair growing. The process is depicted in Fig. 1.

First, we manually define a binary mask on the head mesh that determines the hair and face regions. Note that automatic methods like projecting 2D segmentation could also be used. The mask does not need to be very accurate since it is only used to define the region of the face that we want to preserve and the rough region of hair which will be smoothed out.

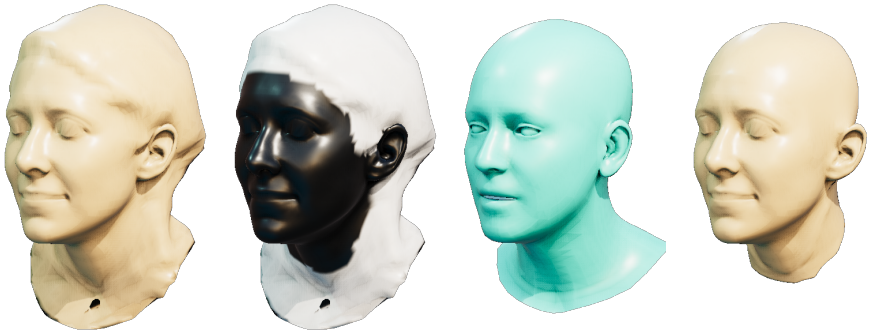
The mesh vertices that fall on the face region are used to fit a FLAME mesh [3]. FLAME provides a low-dimensional space for facial shape, expression and pose. We optimize the FLAME parameters using a Chamfer distance between the face vertices and FLAME template:

$$\mathcal{L}_{face} = \sum_{\mathbf{x} \in X_{face}} \min_{\mathbf{y} \in Y} \|\mathbf{x} - \mathbf{y}\|_2, \tag{1}$$

where X_{face} is the set of 3D points on the facial region of the MVS mesh, and Y are the points of the FLAME template. Note that we only optimize the distance from MVS mesh towards and FLAME mesh, not a bidirectional Chamfer, since we are only interested in accurately matching the facial region, not deforming the template towards the hair.

More robust methods like facial keypoint matching could also be used but we found that since we have a good initial guess for the head position, the model tends to converge to a correct solution with a simple Chamfer distance.

Secondly, we deform the hair region of the original head mesh towards the FLAME mesh, effectively smoothing out the hair region until it only covers the scalp. For this, we deform the hair vertices by using another Chamfer distance



(a) Head mesh (b) Head mask (c) Fitted FLAME mesh (d) Bald head mesh

Fig. 1: Template fitting. Multi-View stereo meshes also the hair region. We fit a FLAME model to the face and deform the hair region in order to recover a plausible scalp geometry.

towards the FLAME mesh together with a Laplacian loss of the hair region to ensure a smooth deformation.

$$\mathcal{L}_{hair} = \sum_{\mathbf{x} \in X_{hair}} \min_{\mathbf{y} \in Y} \|\mathbf{x} - \mathbf{y}\|_2 + \Delta \mathbf{x}. \tag{2}$$

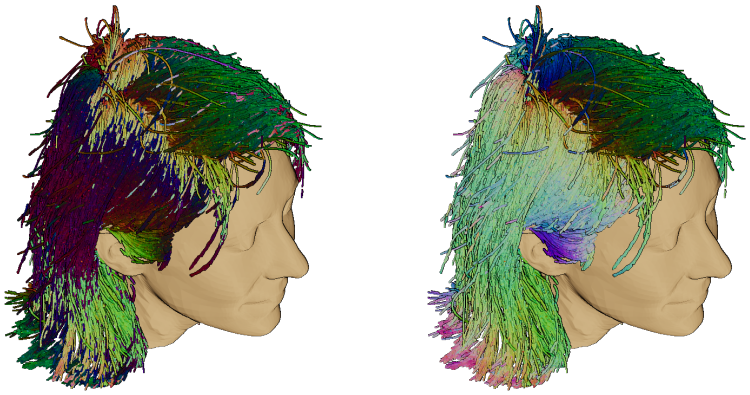
Lastly, we define a UV parametrized scalp mesh on the FLAME mesh which we use as a consistent base for all the real and synthetic subjects.

2 Direction Diffusion

Our strand fitting relies on the Chamfer distance between the generated strands and the LMVS segments. Our Chamfer distance also includes a term on the direction of growth. For this, we require a directional vector for each line-segment that corresponds with the hair growth direction. However, the raw line-segment directions are ambiguous, in that they don't necessarily correspond with the growth direction. To solve this, we require a user to manually select the correct stroke direction for some of the line-segments and we smoothly diffuse this direction until we resolve all the line segments. The diffusion algorithm is similar to the 2D diffusion proposed by Chai *et al.* [1] but we diffuse instead in 3D.

We denote each point from LMVS with $\mathbf{x_i}$ and it's ambiguous direction with $\tilde{\mathbf{d}}_i$. Since the vertex order of the LMVS is arbitrary, the correct hair-growth direction is denoted with $\mathbf{d}_i = s_i \tilde{\mathbf{d}}_i$, where $s_i \in \{-1, 1\}$. The user selects a series of points in the mesh and establishes a *resolved direction* for them: $\mathbf{d}_i = \tilde{\mathbf{d}}_i$.

We define a score P for every point which defines how consistent a certain vertex is w.r.t the neighbouring points with *resolved direction*:



(a) Ambiguous directions (b) Diffused direction

Fig. 2: Direction disambiguation. Line-segments from multi-view stereo have arbitrary direction. We diffuse the hair growth directions from user strokes in order to recover the correct hair growth direction for every line-segment. The per-point directions are visualized as RGB color.

$$P(\mathbf{x}_i) = \sum_{\mathbf{d}_j \in \mathcal{N}_x} \mathbf{d}_i \cdot \mathbf{d}_j, \tag{3}$$

where $\mathcal{N}_x$ denotes the neighbours of point $\mathbf{x}$ which have a *resolved direction*. Resolving the directions of all the points implies the following binary integer problem:

$$\arg\max_{s_i} \sum_i P(\mathbf{x}_i). \tag{4}$$

To solve this, we use a greedy assignment algorithm consisting of three steps:

1. For each unresolved point that has a neighbouring resolved direction, establish a priority of $|P(\mathbf{x}_i)|$ and add it to a priority list,
2. Pop from the unresolved set the point with the highest priority and set s_i so that $P(\mathbf{x}_i)$ is non-negative, and add the point to the resolved set, and
3. Update the priority of the points that are neighbouring the point we have previously resolved.

In Fig. 2, we depict the original ambiguous directions of the line segments and the final diffused ones.

3 Appearance Loss

Using only the geometric loss to fit our strand can sometimes lead to unrealistic reconstruction. We observe that the appearance loss can offer an important supervision signal for the strand fitting, allowing them to converge easier towards

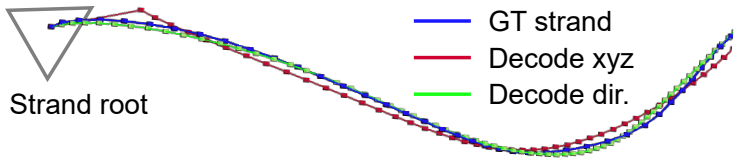


Fig. 4: Gradient integration. Predicting the gradient along the strand instead of decoding directly the 3D position leads to more smooth strands, especially close to the root.

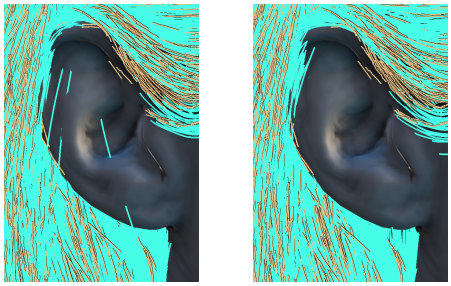
a plausible solution. Fig. 3 shows the impact of disabling the appearance loss when fitting strands.

4 Gradient integration

We experimented with both versions of the decoder, one that predicts the gradients of strands ($\mathcal{G}_{dir}$) and the other that directly predicts 3D positions ($\mathcal{G}_{xyz}$). We observed that $\mathcal{G}_{xyz}$ often generates unnatural strands when fitting to LMVS data; the points near the root can largely deviate from the scalp (see image below).

Our gradient-based strand generator ($\mathcal{G}_{dir}$) effectively solves the problem. We believe that this is due to the nature of smooth gradients of hair strands.

Also, the SIREN architecture in $\mathcal{G}_{dir}$ is well known for its ability to encode first-order derivative information. While in theory $\mathcal{G}_{dir}$ can suffer from the error accumulation problem, we have not observed such an issue in practice.



(a) Without appearance loss

(b) With appearance loss

Fig. 3: Joint optimization. Using only the geometry loss, the hair strand can look unrealistic and penetrate through geometry (blue strands). Using a joint loss of both geometry and appearance helps the strand align better with the line-segments depicted in yellow.

5 Occluded hair

Our method is only supervized in the visible region of the hair with the appear ace loss and the geometric loss towards the sparse hair segments from LMVS. We visualize the hair strands also in the occluded region by masking out one half of the subject with the hair-bun. We observe that the hair strands

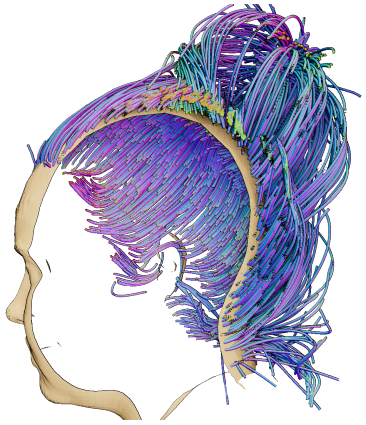


Fig. 5: Occluded hair. Despite being supervised with the visible part of the hair, our strand decoder shows reasonable and smooth hair strands even in occluded regions.

follow a natural and smooth trajectory until they hit the scalp, indicating that the strand generator learns a meaningful prior able to fit even with partial data.

6 Network Architectures

6.1 Strand Generator

We implement the strand generator network as a modulated SIREN [5]. The strand generator takes as input the latent shape vector $\mathbf{z}_g$ of size 64 and a parameter $t \in [0, 1]$ and outputs a direction vector for that strand node. The directions of all the nodes are integrated in order to recover the explicit shape of the strand.

The modulated SIREN consists of two MLP networks: a synthesizer and a modulator. Each has 3 layers of size 32. Each layer of the modulator is element-wise multiplied with the corresponding layer of the synthesizer as per the original architecture of Mehta *et al.* [5]. The weights of the linear layers are reparametrized using weight normalization [6]. The modulator uses the Swish activation and the synthesizer uses a sine activation. The output of the synthesizer is a 3D directional vector. A representation of the strand generator architecture can be seen in Fig. 6.

6.2 Hair Renderer

The hair renderer is implemented as a UNet architecture. The input to the UNet is a screen-space map of 23 channels (16 for the per-strand appearance, 3 for the per-node direction, 1 for the t parameter and 3 for the view direction). The output is an RGBA map of the hair region.

The UNet starts by first mapping the input to 32 channels using a 3×3 convolutional layer. Subsequently, we use 5 downsampling stages which gradually increase the number of channels up to the maximum of 512 at the coarsest level. The decoder mimics the encoder and gradually upsamples and concatenates the corresponding maps from the downsampling stage.

For downsampling and upsampling, we use the approach presented by Karras *et al.* [2] which uses a 6-tap filter. Additionally, we use the filtered leaky ReLU non-linearity [4] which performs an upsampling, activation and downsampling of the input map in order to prevent aliasing. Similar to their architecture, we also limit the cutoff frequency for the coarse maps of the decoder and gradually increase it until reaching the Nyquist limit at the end of the decoder. At the end of the UNet architecture, we use two layers which are critically sampled (filter cutoff is set exactly to the Nyquist limit).

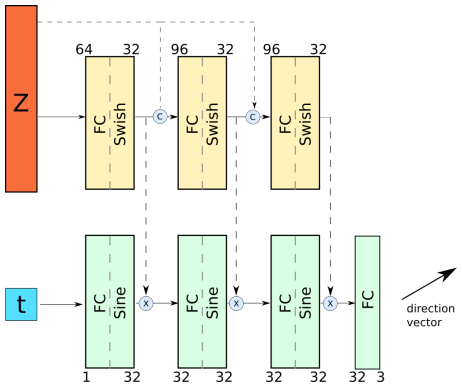


Fig. 6: Modulated SIREN. The latent vector for the strand shape $\mathbf{z}_g$ is concatenated at each layer of the modulator network (top) which produces modulation signals. The synthetizer network is given the t parameter corr responding to the relative position on the strand. The modulation signals are element-wise multiplied with the sine activations from the synthetizer. Finally a fully-connected layer maps to the 3D directional vector.

All layers of the UNet use 3×3 convolutions followed by filtered leaky ReLU. The convolutional layers are reparametrized using weight normalization [6].

7 Training Details

We jointly optimize the parameters of the shape texture $\mathbf{Z}_g$, appearance texture $\mathbf{Z}_a$, and UNet parameters using the geometric loss and the appearance loss. Additionally, we also learn a body and background texture. We use the Adam optimizer and set learning rate 1×10^{-3} for the textures and 1×10^{-4} for the UNet hair renderer. Coarse-to-fine optimization for the shape texture is only performed for the first 10 000 iterations. Root-to-tip is done for the first 1000 iterations.

We create 20 000 strands of 100 points each and their root positions are kept fixed during optimization. The Chamfer distance for the geometric loss is computed between a random batch of 300 000 points from the generated strands and 30 000 points from the line-segments. At each iteration, different batch of random points are chosen from the generated strand and the line-segments to

ensure full coverage. The appearance loss is computed on the full image of size 1024×667

References

1. 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)
2. 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)
3. 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)
4. Maas, A.L., Hannun, A.Y., Ng, A.Y., et al.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. vol. 30, p. 3. Citeseer (2013)
5. 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)
6. Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. Advances in neural information processing systems **29**, 901–909 (2016)