Supplementary Materials for Neural Haircut: Prior-Guided Strand-Based Hair Reconstruction

Vanessa Sklyarova[1] Jenya Chelishev[2,*] Andreea Dogaru[3,*] Igor Medvedev[1]
Victor Lempitsky[4] Egor Zakharov[1]

[1]Samsung AI Center [2]Rockstar Games [3]FAU Erlangen-Nürnberg [4]Cinemersive Labs

1. Implementation and training details

1.1. Datasets preprocessing

We train our hair priors on the publicly available USC-HairSalon [12] synthetic dataset, which consists of 343 hairstyles with up to 10,000 strands aligned with a template bust mesh. Additionally, we match the FLAME head mesh [18] with the template and obtain a UV mapping for the scalp region using Blender[6]. We evaluate our method using real-world H3DS [27] multi-view dataset, monocular videos, and synthetic Cem Yuksel's Hair Models [40].

USC-HairSalon. To train a parametric prior for individual hair strands, we follow [29] approach to pre-processing. We map each strand into a local tangent-bitangent-normal (TBN) basis using the vertices from the closest face to its root location on the FLAME head mesh. While the normal vector in this basis is calculated using the head mesh and is therefore consistent for nearby strands, to ensure consistency in the other two vectors, we orient the tangent vector in a way that aligns with the u direction of the UV texture coordinates map. The bitangent vector is then defined as a cross-product between the normal and the tangent. The origin of this new coordinate system is the strand's root, so after alignment, each strand originates from $\mathbf{0}$.

We increase the diversity of hair strands following [29] and augment their aligned versions using flipping, stretching and squeezing, and rotations around the normal. On top of that, we also add realistic curliness augmentations and cutting into the mix. We apply the same augmentations besides rotations and flipping to the entire hairstyle for the diffusion-based prior training.

H3DS. We evaluate our approach using a public subset of a multi-view H3DS dataset [27]. Each of its scenes has 32 views evenly spaced around the subject. However, since the subject is being moved during the capture because of the non-uniform coverage of the camera setup, their extrinsic parameters are not accurately estimated for some of the scenes. This results in poor performance across all reconstruction methods, and we remove such scenes from the evaluation. We also process these images using human matting [14] and semantic segmentation [21] networks to obtain hair and bust masks. Lastly, we calculate orientation maps using a set of 180 Gabor filters G_b with variances $\sigma_x = 1.8$ and $\sigma_y = 2.4$, frequency $\omega = 0.23$, a zero phase offset ψ, and a rotation angle b, measured in radians. We then obtain an orientation angle a for each pixel: $a = \arg\max_b |G_b|$, and additionally calculate its variance as

$$\mathrm{Var}[a] = \sum_{b\in[0,\pi)} \frac{|G_b|}{\sum_o |G_o|} \cdot \min\{|a-b|^2,\ |a-b\pm\pi|^2\}. \quad (1)$$

Monocular videos. We conduct an additional evaluation of our method by training on monocular videos. For this setup, we place subjects in a chair and ask them to remain stationary during the capture session, which lasts around one minute and is produced using a Samsung Note20 Ultra smartphone. Then, we subsample 60 frames from the video, ensuring that they are equally spaced around the subject and have no motion blur. For that, we use image quality assessment networks [34]. We then perform structure-from-motion using COLMAP [30, 31] to obtain initial values for camera intrinsic and extrinsic parameters. Lastly, we obtain segmentation masks and orientation maps for the training frames following the procedure described previously. We find that additional camera fitting procedure described in [19] launched for first 10,000 iterations during the first stage could improve the quality of reconstructions.

Cem Yuksel's Hair Models. For the quantitative evaluation, we chose two medium-length hairstyles: curly and straight, from a popular synthetic dataset [40]. We used a separate dataset from USC-HairSalon for evaluation to

[*] Work done at Samsung AI Center

avoid bias in the calculated metrics. By using Blender [6], we generate 70 views with a resolution of 2048×2048 for each scene for training, which includes both RGB and segmentation masks. We then calculate the orientation maps using the same procedure based on Gabor filters.

1.2. Hair prior training

Each hair strand in the synthetic dataset is represented as a set of L points: $\mathbf{S} = \{\mathbf{p}_l\}_{l=1}^{L}$, while each hairstyle sample consists of M strands: $\{\mathbf{S}_i\}_{i=1}^{M}$. The number of points per each strand is $L = 100$ across the whole dataset, while the number of strands M varies from sample to sample.

Hair strand parametric model. As stated previously, we map the individual hair strands into a TBN basis and augment them. Then, we encode the aligned 3D points using an encoder $\mathcal{E}$, a one-dimensional ResNet-50 [11], into the mean $\mathbf{z}_\mu \in \mathbb{R}^{64}$ and sigma $\mathbf{z}_\sigma \in \mathbb{R}^{64}$. We then perform a reparameterization trick $\mathbf{z} = \mathbf{z}_\mu + \mathbf{z}_\sigma \cdot \epsilon$ and decode the resulting latent vector into a strand via a decoder $\mathcal{G}$. Instead of predicting individual points, we follow [5] and predict offsets $\mathbf{d}_i^l = \mathbf{p}_i^{l+1} - \mathbf{p}_i^l$. We use a modulated SIREN [23] network consisting of two MLPs with 8 layers and 256 channels for the decoder architecture following [29]. We use this module to individually decode each offset on the strand given its index l and the latent vector $\mathbf{z}$ as inputs. The index l is normalized and used in the periodic activation functions. The resulting points on the strands are obtained by accumulating the offsets:

$$\mathbf{p}_i^l = \sum_{j=1}^{l-1} \mathbf{d}_i^j, \quad l = 2 \dots L, \tag{2}$$

and $\mathbf{p}_i^1 = \mathbf{0}$ due to alignment. The training of $\mathcal{E}$ and $\mathcal{D}$ proceeds using the training objective described in the main paper. For optimization we use Adam [15] with cosine annealing of learning rate from 10^{-4} to 10^{-5} and weights $\lambda_d = 0.05$, $\lambda_c = 1$, $\lambda_{\text{KL}} = 10^{-4}$. After training, the weights of these networks remain frozen, and $\mathbf{z}_\mu$ is used as $\mathbf{z}$.

Hairstyle diffusion model. We prepare a training sample for the diffusion model by first mapping a hairstyle $\{\mathbf{S}_i\}_{i=1}^{M}$ with random strand origins $\mathbf{p}_i^1$ into a hairstyle whose origins span a uniform grid on the FLAME scalp texture map. For that, we use nearest neighbors interpolation. We use the texture with resolution 256×256 for both the hairstyle prior training and fine-tuning. After that, we apply the common augmentations for the entire hairstyle which were described in the previous section in the basis calculated as an average over the basis of its strands components and map them into a latent texture $\mathbf{T} = \{\mathbf{z}_{ij}\}_{i,j=1,1}^{256,256}$ using $\mathcal{E}$. Lastly, we subsample this texture into a low-resolution version $\mathbf{T}_{\text{LR}} \in \mathbb{R}^{32 \times 32}$ using the random integer offsets $s_i \in [0, 7]$ and $s_j \in [0, 7]$. The low-resolution texture can then be obtained as follows:

$$\begin{aligned} \mathbf{T}_{\text{LR}} = \{\mathbf{z}_{ij} \mid\ & i = s_i + 8q, \\ & j = s_j + 8r,\ q, r = 0 \dots 31\}. \end{aligned} \tag{3}$$

Such subsampling allows us to generate exactly $8^2 = 64$ different training samples per hairstyle, boosting the diversity of the dataset and speeding up the training of the prior.

We then follow EDM [13] training pipeline and sample ϵ from a standard normal distribution and a noise level σ from a log-normal distribution with the mean -1.2 and sigma 1.2. We obtain a noised texture $\mathbf{x}$ as:

$$\mathbf{x} = \mathbf{T}_{\text{LR}} + \sigma \cdot \epsilon. \tag{4}$$

Then, for training we use an equivalent simplified version of $\mathcal{L}_{\text{diff}}$:

$$\begin{aligned} \mathcal{L}_{\text{diff}} = \mathbb{E}_{\mathbf{y},\sigma,\epsilon} \Big[\Big\| & \mathcal{F}\big(c_{\text{in}}(\sigma) \cdot \mathbf{x}, c_{\text{noise}}(\sigma)\big) - \\ & \frac{1}{c_{\text{out}}(\sigma)} \big(\mathbf{y} - c_{\text{skip}}(\sigma) \cdot \mathbf{x}\big) \Big\|_2^2 \Big]. \end{aligned} \tag{5}$$

For derivations, please refer to [13]. In Figure 1, we show the samples of a pre-trained diffusion model. These hairstyles look sparse, as they only contain $32^2 = 1024$ strands.

For diffusion model we use UNet architecture from EDM [13] and optimize it using AdamW [22] with learning rate 10^{-4}, $\beta = [0.95, 0.999]$, $\epsilon = 10^{-6}$, and weight decay 10^{-3}. For scheduling, we use an inverse decay learning rate schedule with inverse multiplicative factor = 20000, factor = 1, and warmup = 0.99. All training on synthetic dataset took 2 days on a single NVIDIA RTX 4090.

After training, the diffusion network $\mathcal{F}$ has its weights frozen.

1.3. Coarse volumetric reconstruction

FLAME fitting. For each scene, we fit a FLAME head mesh using keypoint-based objectives. We detect the ground truth keypoints for the face using an OpenPose [3, 4, 32, 37] and Face Alignment [2] detectors and filter out the frames where the face is not visible. Then, we optimize w.r.t. the FLAME shape and pose parameters by minimizing the difference between the projected head mesh keypoints and the detected ones. First, we optimize global rotation, translation, and scale parameters and then additionally fit shape starting from PIXIE [8] initialization and turning on the shape regularization. We use a pipeline similar to DECA [9] for visible keypoints projection and L-BFGS [20] optimizer with a learning rate set to 0.5.

Volumetric reconstruction. To calculate α_i^{hair} and α_i^{bust}, we follow the approach described in NeuS [36] and convert the SDF values $f_{\text{hair}}(\mathbf{x}_i)$ and $f_{\text{bust}}(\mathbf{x}_i)$ into opacities for

Figure 1: Random hairstyles produced by a pre-trained diffusion model. Each sample consists of 1024 individual strands

a given ray $\mathbf{v}_i$. To obtain a set of points $\{\mathbf{x}_i\}_{i=1}^N$ used in ray marching, we apply the iterative importance sampling algorithm from [24] using blended opacities α.

We then use the FLAME mesh to provide additional training signals for the occluded bust regions, which cannot be correctly reconstructed by simply minimizing the difference between the rendered and ground truth colors and silhouettes. Following prior works for fitting SDFs to mesh-based geometry [1, 10, 33], we regularize the implicit SDF to vanish near the surface of the mesh and match its gradients $\nabla_{\mathbf{x}} f_{\text{bust}}(\mathbf{x})$ to the surface normals $\mathbf{n}(\mathbf{x})$ of the closest point on the FLAME mesh. Additionally, we penalize the non-zero hair occupancies α_i^{hair} inside the bust mesh.

We calculate this loss by reusing the points $\mathbf{x}_i$ sampled during ray marching to make the training process more efficient. We split these points into two groups: those who lie inside the volume Ω_{head} bounded by the mesh, and the ones that lie outside: $\Omega_{\text{out}} = \Omega \setminus \Omega_{\text{head}}$. We additionally sample a set of points $\mathbf{x}_i^{\text{head}}$ on the surface of the head mesh, denoted as Ω_0, to evaluate surface-based constraints. The final loss is denoted as $\mathcal{L}_{\text{head}}$:

$$\begin{aligned}\mathcal{L}_{\text{head}} = &\sum_{\mathbf{x}_i^{\text{head}} \in \Omega_0} \left| f_{\text{bust}}(\mathbf{x}_i^{\text{head}}) \right| + \\ &0.1 \cdot \left(1 - \nabla_{\mathbf{x}_i^{\text{head}}} f_{\text{bust}}(\mathbf{x}_i^{\text{head}}) \cdot \mathbf{n}(\mathbf{x}_i^{\text{head}})\right) \\ + &\sum_{\mathbf{x}_i \in \Omega_{\text{out}}} 0.1 \cdot \exp\left(-\gamma \cdot \left| f_{\text{bust}}(\mathbf{x}_i) \right|\right) \\ + &\sum_{\mathbf{x}_i \in \Omega_{\text{head}}} \left| \alpha_i^{\text{hair}} \right|,\end{aligned} \tag{6}$$

where $\cdot$ denotes a dot product, and $\gamma \gg 1$ is a constant.

To calculate an orientation loss, we follow [25] and use Plucker line coordinates [38] to project the orientation field β at point $\mathbf{x}_s$ along the ray $\mathbf{v}$ into the camera $\mathcal{P}$, the projected 2D direction in the camera coordinates is denoted as $\mathbf{L}(\mathbf{x}_s, \beta(\mathbf{x}_s), \mathcal{P})$. Then, we measure the angle $\hat{a}_{\mathbf{v}}$ between the predicted direction and the camera y-axis, which is module π, i.e. in the range $[0, \pi)$. The direction loss between this predicted angle and the ground-truth orientation $a_{\mathbf{v}}$ with its variance $\text{Var}[a_{\mathbf{v}}]$ in the hair region are measured as follows:

$$\mathcal{L}_{\text{dir}} = \sum_{\mathbf{v}} \frac{\mathbf{m}_{\text{hair}}(\mathbf{v})}{\text{Var}^2[a_{\mathbf{v}}]} \min\left\{ |a_{\mathbf{v}} - \hat{a}_{\mathbf{v}}|,\ |a_{\mathbf{v}} - \hat{a}_{\mathbf{v}} \pm \pi| \right\}, \tag{7}$$

where $\mathbf{m}_{\text{hair}}(\mathbf{v})$ denotes a hair mask value at the rendered pixel, corresponding to the ray $\mathbf{v}$, and the sum is across all rays in the batch.

Network architecture. We use a similar network architecture as NeuS [36], which consists of three MLPs to encode SDF f_{hair} for hair geometry, SDF for head f_{bust}, and scene color, respectively. The geometry networks have 8 hidden layers with a hidden size of 256, Softplus with $\beta = 100$ as the activation function, and a skip connection from the input to the fourth hidden layer.

The hair geometry network first transforms the input points via positional encoding with 8 harmonics and then passes them through the MLP to predict their SDF, $f_{\text{hair}} \in \mathbb{R}$, features $l_{\text{hair}} \in \mathbb{R}^{256}$, and orientations $\beta \in \mathbb{R}^3$. The activation of the orientation head is the Tanh function. As the rest of the bust has lower frequency details, the input points of the head geometry network are positionally encoded with 6 harmonics. Similarly, the network predicts the bust SDF $f_{\text{bust}} \in \mathbb{R}$ and feature vectors $l_{\text{bust}} \in \mathbb{R}^{256}$.

We have a joint color network, which is modeled by an MLP with 4 linear layers with a hidden size of 256. As input it takes the spatial location x_i, the view direction $\mathbf{v}$, the normal vector of SDF, $\mathbf{n} = \nabla f(x_i)$, and a 256-dimensional feature vector $\mathbf{l}$. To combine the feature vectors and normals of the hair, $l_{\text{hair}}, \nabla f_{\text{hair}}(x_i)$, with the bust, $l_{\text{bust}}, \nabla f_{\text{bust}}(x_i)$, we first calculate the blending weight w_i for each point as follows:

$$w_i = \frac{\alpha_i^{\text{bust}}}{\alpha_i^{\text{bust}} + \alpha_i^{\text{hair}} + \varepsilon}, \tag{8}$$

where $\alpha_i^{\text{bust}}, \alpha_i^{\text{hair}}$ - are individual opacities of bust and hair correspondingly and $\varepsilon = 10^{-5}$ is used for numerical stability. Then we blend the features and the normals accordingly:

$$\mathbf{l} = w_i \cdot l_{\text{bust}} + (1 - w_i) \cdot l_{\text{hair}} \tag{9}$$

$$\mathbf{n} = w_i \cdot \nabla f_{\text{bust}}(x_i) + (1 - w_i) \cdot \nabla f_{\text{hair}}(x_i) \tag{10}$$

We train volumetric reconstruction using Adam [15] optimizer with learning rate equal to $5 \cdot 10^{-4}$ and weights: $\lambda_{\text{color}} = 1$, $\lambda_{\text{mask}} = 0.1$, $\lambda_{\text{reg}} = 0.1$, $\lambda_{\text{head}} = 0.1$, $\lambda_{\text{dir}} = 0.1$ for 300,000 iterations.

1.4. Fine strand-based reconstruction

For fine strand-based optimization we use Adam [15] with learning rate set to 10^{-3} and MultiStep annealing with $\gamma = 0.5$ and milestones = $[4 \cdot 10^4, 6 \cdot 10^4, 8 \cdot 10^4]$. Also, we use the following weights of losses: $\lambda_{\text{chm.}} = 1.$, $\lambda_{\text{orient}} = 0.01$, $\lambda_{\text{prior}} = 10^{-3}$, $\lambda_{\text{render}} = 10^{-3}$, $\lambda_{\text{mask}} = 0.01$.

Texture parametrization. Our geometry $\mathbf{T}$ and appearance texture have resolution 256×256 with number of channels 64 and 16 correspondingly. They are both parameterized using a UNet, similar to deep image prior [35]. We share network parameters between these textures and predict them from a constant grid of UV coordinates. We preprocess them using a positional encoding [24] before feeding into the network, which consists of 6 sine and cosine functions.

Visible surface extraction. We obtain the visible hair surface $\mathcal{S}$ from a hair SDF f_{hair} and a bust SDF f_{bust} to use it in orientation $\mathcal{L}_{\text{orient}}$ and chamfer $\mathcal{L}_{\text{chm.}}$ losses. To obtain it, we first extract zero-level iso-surfaces from both implicit functions using Marching Cubes [17]. Then, we render both hair and bust meshes using a set of cameras from the chosen dataset. Due to limited top views in H3DS [27] we additionally consider top cameras to prevent the appearance of big holes in hair SDF geometry. Finally, we select all the hair faces that are visible from at least one view and use the resulting mesh as $\mathcal{S}$ in all losses.

Soft rendering. For differentiable soft rasterization, we use Pytorch3D [28] framework. Our full soft rasterization pipeline consists of three steps. First, we generate quad geometry for each strand in a hairstyle and orient these polygonal quads so that most of the produced faces are aligned with the camera plane, see Fig. 2. It requires calculating Frenet–Serret frame [7] for each point of a strand and then generating additional vertices while considering only XY coordinates in camera space:

$$\mathbf{S}_{\text{gen.}} = \mathbf{S} \pm [\mathbf{N}_{XY}, 0], \tag{11}$$

where $\mathbf{S}$ – hair strand represented by 3D vertices, $\mathbf{N}_{XY}$ – normal vector to a strand projection onto a camera plane, by $[\cdot, \cdot]$ we denote a simple concatenation. Such view-aware

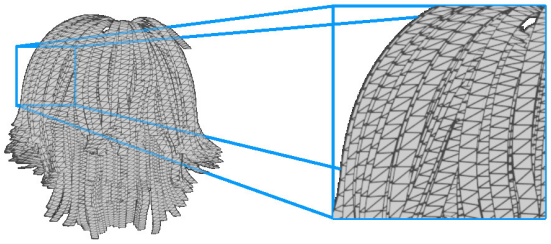

Figure 2: Hair quads produced by the view-aware generation: geometry is built in a way that most of the quads are facing the camera plane.

generation prevents quads from being oriented orthogonal to the view plane, which effectively increases the number of samples as if they were oriented randomly.

Secondly, we rasterize obtained quads for hair within head to obtain z-buffer with the nearest faces to each pixel. Finally, we blend it using sigmoid probability map. For rasterization we use blur radius = 10^{-4}, faces per pixel = 16 and image size = 512 due to memory restrictions and for blending: $\sigma = 10^{-5}$ and $\gamma = 10^{-5}$.

The input of our rendering UNet network consists of soft rasterized appearance descriptors $\in \mathbb{R}^{16}$ concatenated with hard rasterized orientations $\hat{a}$ transformed to $\mathbb{R}^3$ using sine and cosine functions. For the cosine, we additionally split it into two channels, corresponding to the positive and negative components, and take their absolute value. This way, we ensure that all channels are normalized in $[0, 1]$. While the rasterized appearance features are the same for the whole strand, orientations are different for each point. They contribute to the ability of neural rendering to model the view-dependent changes in hair color since the projected orientations contain information about both hair strand local growth direction and camera view direction. For rendering UNet, we use architecture similar to [26].

2. Additional Ablations and Results

Evaluation protocol. For metrics evaluation, we sample 50,000 strands, the same number as in ground-truth hairstyle. Furthermore, we linearly interpolate the number of points on each strand in ground-truth hairstyle to 100 to prevent biased metrics.

Real-world evaluation. We show an extended comparison of our approach with reconstruction methods for different viewing angles (see Figure 5) on H3DS [27]. Furthermore, we visualize the reconstructions for additional scenes in Figure 6. Our approach can handle both long and short hairstyles, proving its versatility in achieving realistic re-

Method	Straight hair									Curly hair								
	2/20	3/30	4/40	2/20	3/30	4/40	2/20	3/30	4/40	2/20	3/30	4/40	2/20	3/30	4/40	2/20	3/30	4/40
	Precision			Recall			F-score			Precision			Recall			F-score		
$\mathcal{L}_{\text{geom}}$	63.8	88.6	94.9	9.9	16.2	21.2	17.1	27.4	34.7	50.8	75.1	85.9	5.7	11.3	18.4	10.2	19.6	30.3
w/o $\mathcal{L}_{\text{chm}}$	82.9	95.0	97.1	4.5	8.9	14.2	8.5	16.3	24.8	51.0	73.8	84.6	3.9	8.4	14.3	7.2	15.1	24.5
w/o $\mathcal{L}_{\text{vol}}$	48.3	71.8	79.4	10.1	21.7	32.2	16.7	33.3	45.8	20.1	35.3	45.5	5.7	12.4	21.2	8.9	18.4	28.9
w/o $\mathcal{L}_{\text{orient}}$	31.7	56.2	69.0	6.0	12.1	17.8	10.1	19.9	28.3	21.5	43.7	59.8	4.7	10.3	17.7	7.7	16.7	27.3
w/ $\mathcal{L}_{\text{render}}$ [29]	68.4	89.4	95	9.8	15.7	23.6	17.1	26.7	37.8	48.7	75.3	87.0	6.2	12.0	19.3	11.0	20.7	31.6
w/ $\mathcal{L}_{\text{rgb}}$	71.6	90.4	95.2	9.1	15.6	22.5	16.1	26.6	36.4	49.3	76.0	87.7	6.1	12.0	19.4	10.9	20.7	31.8
w/ $\mathcal{L}_{\text{mask}}$	63.5	88.2	94.6	11.1	17.3	22.5	18.9	28.9	36.4	49.4	74.7	86.1	6.3	12.1	19.5	11.2	12.1	31.8
$\mathcal{L}_{\text{fine}}$ w/o $\mathcal{L}_{\text{rgb}}$	59.8	84.1	92.2	12.9	22.8	31.3	21.2	35.9	46.7	45.1	71.1	83.6	6.3	12.4	20.3	11.1	21.1	32.7
$\mathcal{L}_{\text{fine}}$	59.9	84.1	92.1	13.1	22.7	31.5	21.5	35.8	46.9	45.8	72.1	84.6	6.4	12.8	21.0	11.2	21.7	33.6
Neural Strands* [29]	74.0	81.8	85.3	12.8	20.5	28.8	21.8	32.8	43.1	38.4	59.8	72.4	7.9	15.1	23.8	13.1	24.1	35.8

Table 1: We provide an extended quantitative evaluation of individual components of our method with per-scene metrics. Our full method with $\mathcal{L}_{\text{fine}}$ outperforms others in terms of Recall and F-score for both scenes. For a detailed discussion, please refer to Section 2.

Method	2/20	3/30	4/40	2/20	3/30	4/40	2/20	3/30	4/40
	Precision			Recall			F-score		
final model	45.8	72.1	84.6	6.4	12.8	21.0	11.2	21.7	33.6
32 faces per pixel	42.8	67.8	81.4	5.8	11.7	19.5	10.2	20.0	31.5
1 face per pixel	42.4	69.5	83.7	5.9	12.2	20.7	10.4	20.8	33.2
batch size 4	38.8	66.6	81.9	6.2	12.6	20.8	10.7	21.2	33.2
batch size 8	43.7	67.9	81.4	6.3	12.8	21.1	11.0	21.5	33.5

Table 2: We provide an extended quantitative evaluation of hyperparameters of our method. Our final model outperforms others, showing that its set of hyperparameters is optimal.

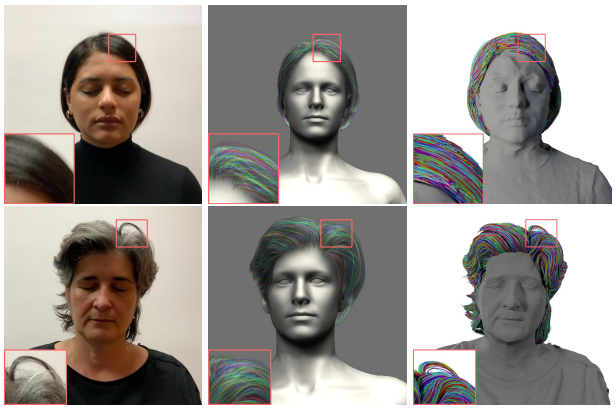

Figure 3: Comparison of our multi-view method (right) with a single-shot NeuralHDHair [39] system (middle). Digital zoom-in is recommended.

constructions in different scenarios. Finally, we provide results of our system obtained on the same twelve views as used in DeepMVSHair [16] (see Figure 11).

We also provide an extended comparison with the one-shot reconstruction method NeuralHDHair[39], see Figure 3. The main advantage of our method is the higher fidelity of hair reconstructions, which are obtained jointly with the personalized bust models.

Lastly, we include additional monocular video reconstruction examples. In total, we present the results for five scenes with various hairstyles: long, short, and curly, see Figure 7, 8 and 9.

Ablation on losses. We provide an extended ablation study in Table 1 and Figure 4. It contains separate results for curly and straight synthetic hair, which extend quantitative metrics presented in the main paper. We additionally include an ablation study for the individual components of the geometry loss $\mathcal{L}_{\text{geom}}$ in the upper section and the importance of $\mathcal{L}_{\text{rgb}}$ in the bottom. Our full method, $\mathcal{L}_{\text{fine}}$, achieves the best performance in terms of both Recall and aggregated F-score for both scenes. All terms in $\mathcal{L}_{\text{geom}}$ clearly contribute to the overall performance. Without $\mathcal{L}_{\text{chm}}$ the generated hairstyle does not cover the whole hair volume. Since we use only a one-way chamfer, not two-way, to attract strands to the outer surface, discarding $\mathcal{L}_{\text{vol}}$ leads to unrealistic strands outside the hair region (see Figure 4). Without $\mathcal{L}_{\text{orient}}$ we obtain the same hairstyle coverage, but strands have random orientations on the surface which significantly decreases the realism. Furthermore, from Table 1 you could see the decrease in performance for both scenes without using $\mathcal{L}_{\text{rgb}}$. We also present the comparison results with Neural Strands* re-implementation in Table 1 (last row). Our method outperforms Neural Strands [29] both quantitatively (across most metrics) and qualitatively as was shown in the main paper.

Hyperparameters study. We conduct an ablation study on important hyperparameters used in the soft rendering part in order to make sure that the chosen ones are optimal. Results on curly synthetic hair scene are provided in Table 2. We varied different number of faces per pixel and images used at each iteration in soft rasterization. In our final model by default, we use faces per pixel = 16 and batch size = 1. From the table, we could see that neither increase nor decrease of these hyperparameters helps to improve results compared to our final model. The quality of renders is also the same.

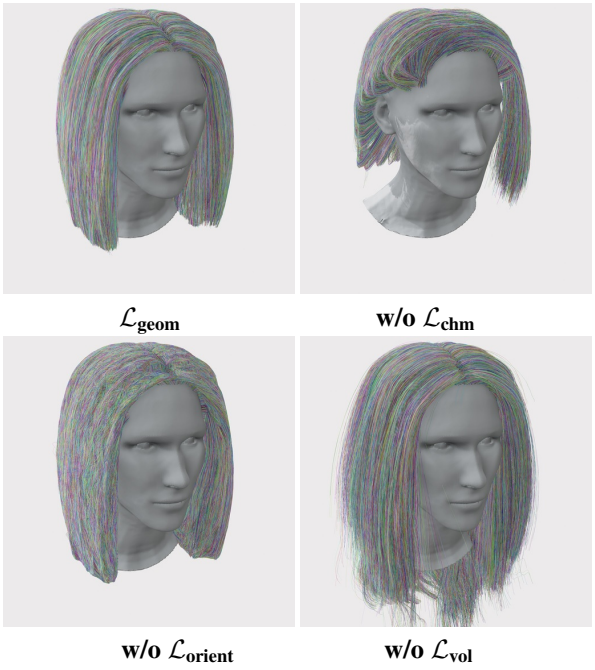

Figure 4: Ablation on individual components of geometry loss $\mathcal{L}_{\text{geom}}$. Without chamfer loss $\mathcal{L}_{\text{chm}}$ strands doesn't cover the whole hair silhouette. Removing orientation loss $\mathcal{L}_{\text{orient}}$ leads to random directions while removing the volume loss $\mathcal{L}_{\text{vol}}$ results in uncontrolled strands growing outside the hair region.

Figure 5: Extended qualitative comparison using real-world multi-view scenes [27]. Digital zoom-in is recommended.

Figure 6: Additional results for our method on a multi-view dataset [27]. Our method is capable of reconstructing various length hairstyles, starting from long (top row) to short (bottom row). Digital zoom-in is recommended.

Figure 7: Additional reconstruction results of our method on monocular videos in arbitrary lighting conditions. Our method is capable of obtaining personalized reconstructions for various hairstyles.

Figure 8: Additional reconstruction results of our method on monocular videos in arbitrary lighting conditions. Our method could produce realistic hair geometry for long and short, straight and curly hairstyles.

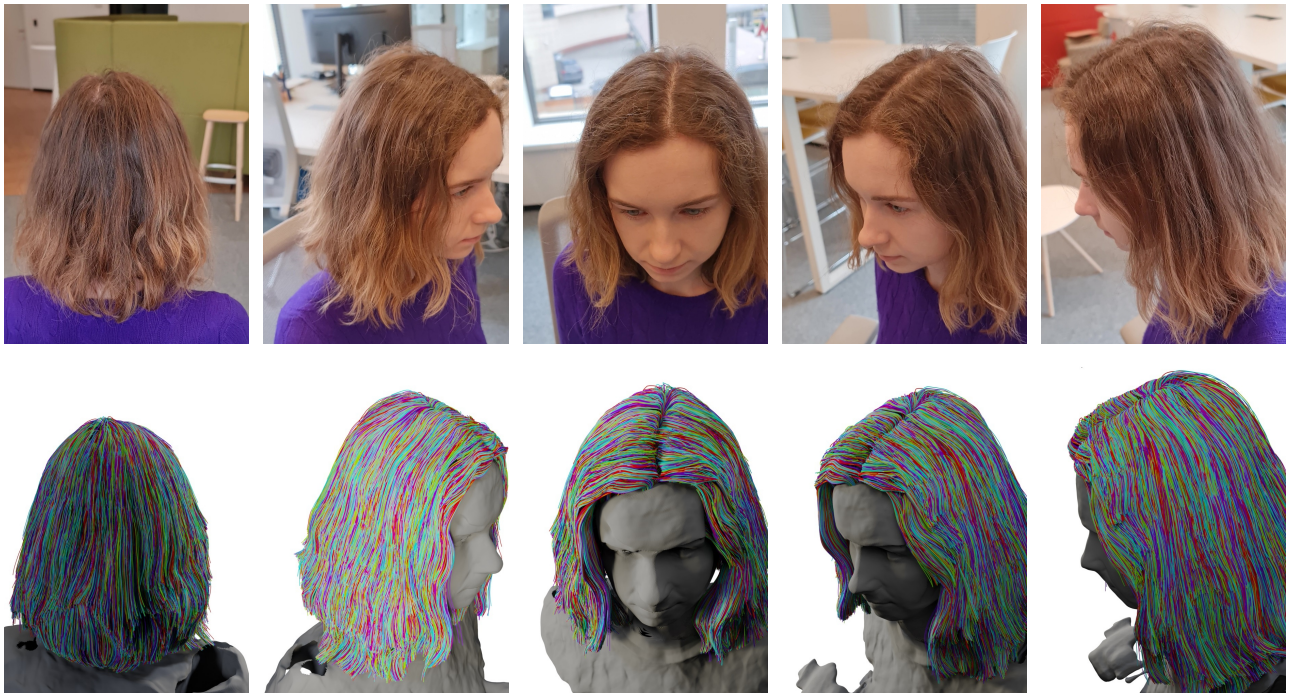

Figure 9: Additional reconstruction results of our method on monocular videos in arbitrary lighting conditions.

Figure 10: The main limitation of our method is related to curly hair reconstruction, which will be addressed in future work.

Figure 11: Qualitative comparison using 12 views from real-world multi-view scenes [27]. Digital zoom-in is recommended.

References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2562–2571, 2019. 3

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017. 2

[3] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 2

[5] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2018. 2

[6] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2023. 1, 2

[7] Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. Digital geometry processing with discrete exterior calculus. In *International Conference on Computer Graphics and Interactive Techniques*, 2013. 4

[8] Yao Feng, Vasileios Choutas, Timo Bolkart, Dimitrios Tzionas, and Michael Black. Collaborative regression of expressive bodies using moderation. In *International Conference on 3D Vision (3DV)*, pages 792–804, Dec. 2021. 2

[9] Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. volume 40, 2021. 2

[10] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, 2020. 3

[11] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 2

[12] Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. Single-view hair modeling using a hairstyle database. *ACM Transactions on Graphics (TOG)*, 34:1 – 9, 2015. 1

[13] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[14] Zhanghan Ke, Jiayu Sun, Kaican Li, Qiong Yan, and Rynson W.H. Lau. Modnet: Real-time trimap-free portrait matting via objective decomposition. In *AAAI*, 2022. 1

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 2, 4

[16] Zhiyi Kuang, Yiyang Chen, Hongbo Fu, Kun Zhou, and Youyi Zheng. Deepmvshair: Deep hair modeling from sparse views. *SIGGRAPH Asia 2022 Conference Papers*, 2022. 5

[17] Thomas Lewiner, Hélio Lopes, Antônio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8:1 – 15, 2003. 4

[18] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. 1

[19] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 1

[20] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989. 2

[21] Kunliang Liu, Ouk Choi, Jianming Wang, and Wonjun Hwang. Cdgnet: Class distribution guided network for human parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4473–4482, June 2022. 1

[22] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. 2

[23] Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated periodic activations for generalizable local functional representations. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14194–14203, 2021. 2

[24] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 2020. 3, 4

[25] Giljoo Nam, Chenglei Wu, Min H. Kim, and Yaser Sheikh. Strand-accurate multi-view hair capture. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 155–164, 2019. 3

[26] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15969–15979, June 2022. 4

[27] Eduard Ramon, Gil Triginer, Janna Escur, Albert Pumarola, Jaime Garcia Giraldez, Xavier Giró i Nieto, and Francesc Moreno-Noguer. H3d-net: Few-shot high-fidelity 3d head reconstruction. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5600–5609, 2021. 1, 4, 7, 8, 12

[28] Nikhila Ravi, Jeremy Reizenstein, David Novotný, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *SIGGRAPH Asia 2020 Courses*, 2019. 4

[29] Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. Neural strands: Learning hair geometry and appearance from multi-view images. *European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 5

[30] Johannes Lutz Schönberger and Jan-Michael Frahm.

Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[31] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1

[32] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017. 2

[33] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[34] Shaolin Su, Qingsen Yan, Yu Zhu, Cheng Zhang, Xin Ge, Jinqiu Sun, and Yanning Zhang. Blindly assess image quality in the wild guided by a self-adaptive hyper network. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3664–3673, 2020. 1

[35] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. *Int. J. Comput. Vis.*, 128(7):1867–1888, 2020. 4

[36] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3

[37] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016. 2

[38] Bernhard P. Wrobel. Multiple view geometry in computer vision. *Künstliche Intell.*, 15:41, 2001. 3

[39] Keyu Wu, Yifan Ye, Lingchen Yang, Hongbo Fu, Kun Zhou, and Youyi Zheng. Neuralhdhair: Automatic high-fidelity hair modeling from a single image using implicit neural representations. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1516–1525, 2022. 5

[40] Cem Yuksel, Scott Schaefer, and John Keyser. Hair meshes. *ACM SIGGRAPH Asia 2009 papers*, 2009. 1