PERM: A Parametric Representation for Multi-Style 3D Hair Modeling – Supplementary Material –

CHENGAN HE, Yale University, USA
XIN SUN, Adobe Research, USA
ZHIXIN SHU, Adobe Research, USA
FUJUN LUAN, Adobe Research, USA
SÖREN PIRK, CAU, Germany
JORGE ALEJANDRO AMADOR HERRERA, KAUST, Saudi Arabia
DOMINIK L. MICHELS, KAUST, Saudi Arabia
TUANFENG Y. WANG, Adobe Research, USA
MENG ZHANG, Nanjing University of Science and Technology, China
HOLLY RUSHMEIER, Yale University, USA
YI ZHOU, Adobe Research, USA

A FORMULATION DETAILS

A.1 Hair Geometry Textures

Unlike triangle meshes that share the same topology between different bodies and faces, hair tends to have a different number of strands for different hairstyles, making traditional PCA-based blend shapes impossible on these data. Although we can force different hairstyles to have the same number of strands by introducing a resampling step, the computed blend shapes would still have a fixed number of strands, which are less flexible in real applications. For flexibility, we store each hair model as a 2D texture map of strand PCA coefficients. With the scalp parameterization proposed by Wang et al. [2009], we are able to unwarp the 3D scalp surface to a 2D *uv* plane. However, naively storing strand PCA coefficients on the projected 2D root positions will cause two problems: (1) different strands may be projected to the same texel due to discreterization, thus causing collision problems; (2) some texels may receive no strands, thus leaving missing values in the projected textures.

To address these issues, we fit our geometry textures with two steps: first, we find the nearest 2D hair root for each texel, and store the corresponding strand PCA coefficients at that texel. For those texels whose distance to its nearest hair root is above a threshold $\epsilon = 0.01$, we store a special vector that will be decoded to strands with zero length, and mark those texels as a baldness map $\mathbf{M}$ [Zhou et al. 2023]. This step ensures no missing values in the texture, which we denote as the initialized geometry texture $\mathbf{T}_{\text{init}}$. Both $\mathbf{M}$ and $\mathbf{T}_{\text{init}}$ have the resolution 256×256, which is empirically set considering the trade-off between size and expressiveness. To ensure that geometry textures can properly recover the original 3D hairstyle, we optimize them directly in the second step:

$$\mathbf{T}^* := \arg\min_{\mathbf{T}} \mathcal{L}_{\text{geo}}\Big(\mathcal{S}\big(\text{Sample}(\mathcal{R};\mathbf{T})\big)\Big), \tag{1}$$

where we use loss $\mathcal{L}_{\text{geo}}$ to measure the reconstruction difference on strand geometry.

We employ the Adam optimizer [Kingma and Ba 2014] with a learning rate of 0.001. With $\mathbf{T}_{\text{init}}$ as initialization, our experiments show that the optimization process converges within 500 iterations, and in Fig. 1 we show an example of the original hairstyle, the fitted geometry texture, and the recovered 3D hairstyles by sampling and decoding from the texture with different numbers of roots. These fitted geometry textures finally form a unified representation across different hairstyles, which have the same size $256 \times 256 \times 64$ and allow for arbitrary sampling.

GT (10K strands) Sample with 10K roots Sample with 100K roots

Fig. 1. Illustration of fitted hair geometry textures and the corresponding 3D hairstyle. Here baldness maps are visualized as the alpha channel in the RGBA textures.

A.2 Network Architectures

StyleGAN2 Backbone. Our StyleGAN2 backbone follows the official implementation of [Karras et al. 2020][1], with a mapping network of 4 hidden layers. We modify the output convolutions such that they produce a feature image of shape $32 \times 32 \times 10$. Subsequently, a small MLP decoder is employed to map the output features to 10-dimensional strand PCA coefficients and a single scalar for guide mask. The MLP decoder consists of a single hidden layer of 64 hidden units and uses the softplus activation function. Note that we

[1] https://github.com/NVlabs/stylegan2-ada-pytorch

do not utilize pre-trained StyleGAN2 checkpoints for our task, the entire module is trained from scratch.

U-Net Super Resolution. Our U-Net module is implemented based on an unofficial online implementation[2], which translates the bilinearly upsampled textures of shape $256 \times 256 \times 11$ to weight maps of shape $256 \times 256 \times 14$. The convolution layers progressively downsample the input to a shape of $16 \times 16 \times 512$, which is followed by 4 bilinear upsampling and double convolution layers with skip connections to produce the weight map.

VAE. In our VAE module, the encoder adopts a similar architecture to pSp [Richardson et al. 2021], which contains 4 IR-SE blocks [Hu et al. 2018] to extract a feature image of shape $4 \times 4 \times 512$. This feature image is then flattened and processed through a fully-connected layer to derive the mean and variance of $\vec{\beta}$ of 512 dimensions. The decoder mirrors the StyleGAN2 generator but omits its mapping network. Its output size is modified to $256 \times 256 \times 54$, followed by the same MLP decoder that maps the output features to 54-dimensional strand PCA coefficients and a single scalar for baldness map.

B EXPERIMENT DETAILS

B.1 Datasets

We train PERM on USC-HairSalon [Hu et al. 2015], which is a dataset comprising 343 3D hair models collected from online game communities. To increase diversity, we employ the data augmentation method proposed in HairNet [Zhou et al. 2018], where different hair models within the same style class are blended to produce novel hairstyles. The blended hairstyles are further augmented by horizontal flipping, resulting in a total of 21,054 data samples used for training.

To assess the performance of our model, we compiled a dataset of 3D hair models from various publicly available resources, including CT2Hair [Shen et al. 2023] (10 hairstyles), StructureAwareHair [Luo et al. 2013] (3 hairstyles), and Cem Yuksel's website[3] (4 hairstyles). We preprocess these data to have the same number of points ($L = 100$) on each strand, and register them onto the same head mesh. For a more comprehensive evaluation, we engaged artists to groom a collection of 100 diverse hairstyles. While these artist-created data will be used for evaluation, we are restricted in releasing them due to regulatory considerations.

B.2 Training Details

We train our model and conduct all experiments on a desktop machine with an Intel® Core™ i9-10850K CPU @ 3.60GHz, 64GB memory, and an NVIDIA RTX 3090 GPU. Our code is implemented with Python 3.9.18, PyTorch 1.11.0, and CUDA Toolkit 11.3.

In our model, each network module is trained separately using the Adam optimizer [Kingma and Ba 2014]. The StyleGAN2 backbone has a learning rate of 0.002 for its generator and 0.001 for its discriminator, leading to a stable training configuration in our case. The StyleGAN2 backbone is trained for 3,000K images with a batch size of 4, which takes around 1 day on our machine. For both the U-Net and VAE, we set their learning rates to 0.002 and train them for 2,000K images with a batch size of 4, each taking around 1 day on our machine.

B.3 Quantitative Metrics

To quantitatively measure the reconstruction capability of our model, we first report the mean *position error* (pos. err.), which is essentially the average Euclidean distance between corresponding points on the reconstructed strands and the ground truth. We further report the mean *curvature error* (cur. err.) that measures the L_1 norm between the curvatures of reconstructed and ground truth strands, where the curvature is defined as the reciprocal of the circumradius of 3 consecutive points $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, and $\mathbf{p}_{i+1}$ on the strand, which can be computed as:

$$\mathrm{cur}(\mathbf{p}_i) = \frac{2\|(\mathbf{p}_{i-1} - \mathbf{p}_{i+1}) \times (\mathbf{p}_i - \mathbf{p}_{i+1})\|}{\|\mathbf{p}_{i-1} - \mathbf{p}_{i+1}\| \cdot \|\mathbf{p}_i - \mathbf{p}_{i+1}\| \cdot \|\mathbf{p}_{i-1} - \mathbf{p}_i\|}. \tag{2}$$

B.4 PCA-based Strand Representation

In Fig. 2a, we illustrate the explained cumulative relative variance against the number of principal components. Although 20 PCA coefficients appear to capture nearly 100% of the variance in the *training set*, increasing the number of coefficients improves the generalizability of our representation to unseen data, as evidenced by the reconstruction errors shown in Fig. 2b on the *testing set*. Considering this issue, we opt for 64 principal components, a choice consistent with most previous work on strand representation [Rosu et al. 2022; Sklyarova et al. 2023; Zhou et al. 2023], while achieving significantly lower reconstruction errors.

B.5 Baselines

GroomGen [Zhou et al. 2023]. As the authors of GroomGen have not released their code, we implement GroomGen by ourselves with Python 3.9.18 and Pytorch 1.11.0. The model is trained on the same USC-HairSalon dataset as described in Sec. B.1.

HairStep [Zheng et al. 2023]. We use the pre-trained HairStep model released by the authors[4].

Strand VAEs. The architecture of different strand VAEs used in our evaluation is adapted from GroomGen [Zhou et al. 2023], where we only modify the output layer to generate either strand directions (Pos. VAE) or frequency components (Freq. VAE).

C ADDITIONAL RESULTS

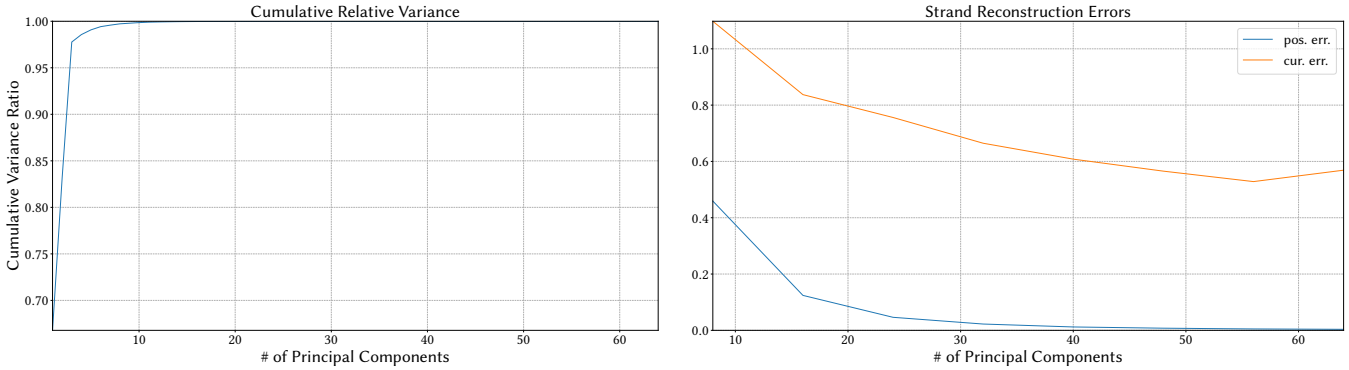
C.1 Strand Representation Comparison

In Fig. 3 we provide an additional comparison of different strand representations. Their configurations are the same as described in our main paper.

C.2 Random Hairstyle Synthesis

In Fig. 4, we showcase several random guide strands generated by sampling the parameter space of $\vec{\theta}$ with Gaussian noise, highlighting that the results from StyleGAN2 exhibit greater diversity compared to those generated by the PCA alternative discussed in our main

[2] https://github.com/milesial/Pytorch-UNet
[3] http://www.cemyuksel.com/research/hairmodels/
[4] https://github.com/GAP-LAB-CUHK-SZ/HairStep



(a) Cumulative relative variance as a function of the number of principal components.

(b) Strand reconstruction errors as a function of the number of principal components.

Fig. 2. Performance of principal components in the training and testing sets.

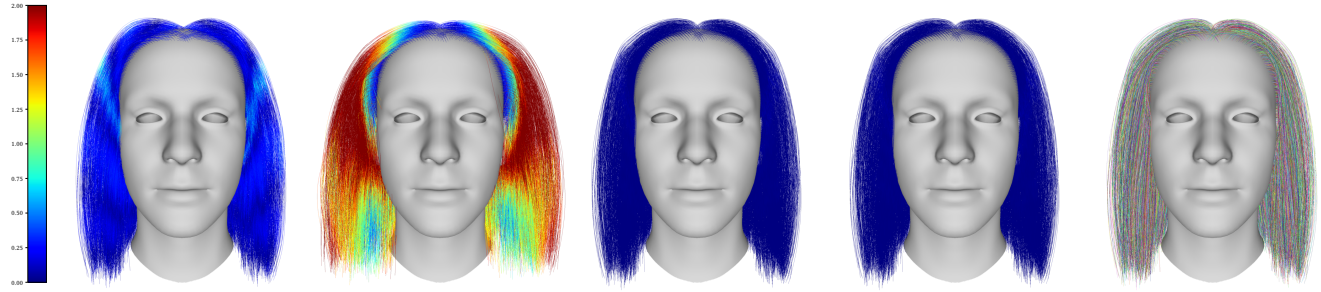


Fig. 3. Comparison of our PCA-based strand representation (Freq. PCA) with other VAE-based representations (Pos. VAE [Rosu et al. 2022; Sklyarova et al. 2023], Freq. VAE [Zhou et al. 2023]) and a simpler PCA-based formulation in the spatial domain (Pos. PCA). Reconstructed strands are color-coded by their position error.

paper. Given that PCA lacks constraints on the distribution of its subspace, obtaining reasonable guide strands by sampling its subspace with Gaussian noise is challenging, and our results indicate that most of them are collapsed into similar outputs.

In Fig. 5, we illustrate several random full hair models generated by sampling the parameter spaces of $\vec{\theta}$ and $\vec{\beta}$ with Gaussian noise, and compare these results to GroomGen [Zhou et al. 2023]. Note that we sample our parameter space and GroomGen's latent space with the same Gaussian noise for a fair comparison. From our observation, GroomGen tends to generate hairstyles with weird curls and flyaway strands, which do not exist in our results. Note that the quality difference from GroomGen's paper may be partially correlated with the dataset difference, as GroomGen in its original paper was trained on their private GROOMHAIR dataset.

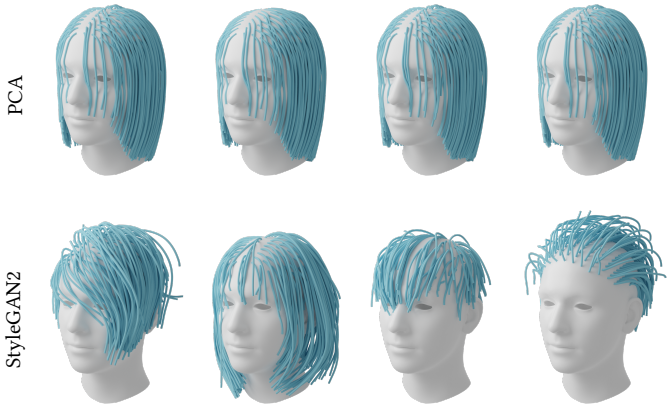


Fig. 4. Guide Strands synthesized from different Gaussian noise (Top: PCA; Bottom: StyleGAN2).

C.3 3D Hair Parameterization

To fit PERM parameters to target 3D hair models, we formulate it as an optimization problem, where the objective is defined as:

$$\vec{\theta}^*, \vec{\beta}^* := \arg\min_{\vec{\theta},\vec{\beta}} \|\mathcal{F}(\mathcal{G}(\vec{\theta})) \oplus \mathcal{D}(\vec{\beta}) - \mathbf{T}\|_1 + \mathcal{L}_{\text{geo}}. \tag{3}$$

We employ the Adam optimizer [Kingma and Ba 2014] with an initial learning rate of 0.1 and a cosine annealing schedule for the learning rate. For better convergence, we first optimize $\vec{\theta}$ only for

Fig. 5. Full hair models synthesized from different Gaussian noise, with comparison to GroomGen [Zhou et al. 2023] trained on the same dataset. Hair colors are manually assigned for aesthetic purposes.

1,000 iterations as a warm-up to match the global shape, and then jointly optimize $\vec{\theta}$ and $\vec{\beta}$ for 4,000 iterations.

With Perm, we fit parameters to hundreds of publicly available 3D hair models sourced from the Internet, with a subset of them showcased in Fig. 6. Similar to AMASS [Mahmood et al. 2019], we curated a dataset of 3D hair in a unified and parametric manner, which we will release to facilitate future research.

C.4 Hairstyle Interpolation

In this section we compare our method with [Weng et al. 2013] and [Zhou et al. 2018] on hairstyle interpolation. Since neither

Fig. 6. A subset of 3D hair models fitted by PERM.

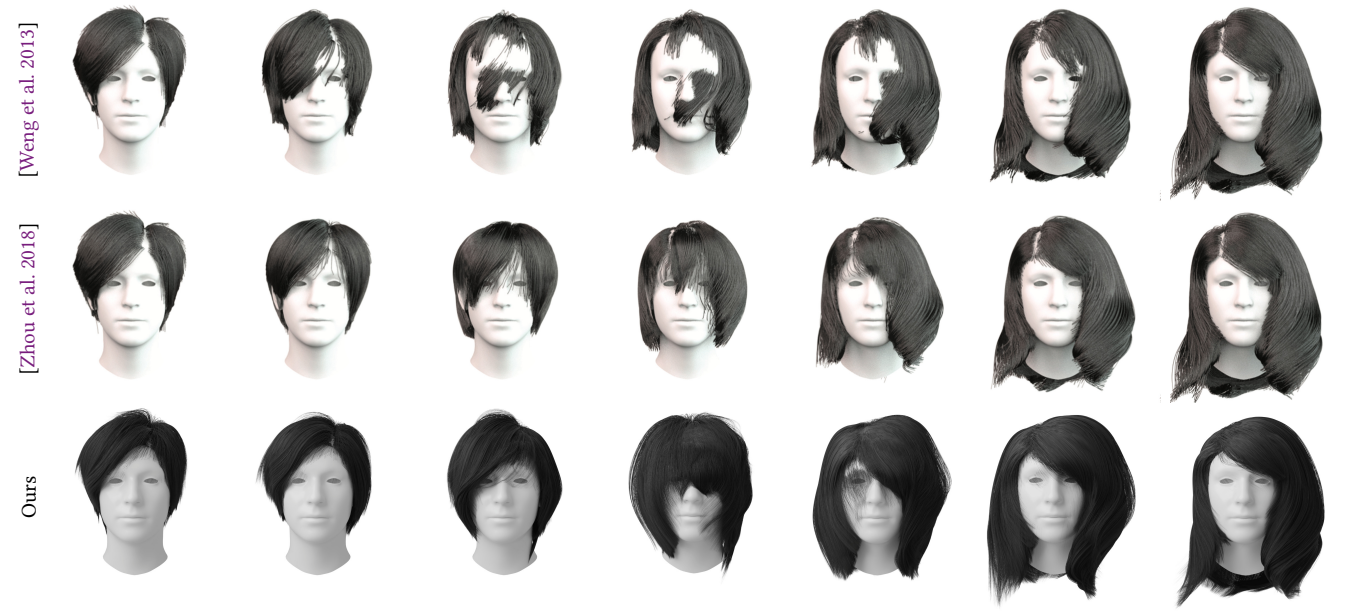


Fig. 7. Interpolation comparison with [Weng et al. 2013] and [Zhou et al. 2018].

[Weng et al. 2013] nor [Zhou et al. 2018] are publicly available, we embed the 2 hairstyles selected by [Zhou et al. 2018] into our parameter space and jointly interpolate the projected $\vec{\theta}$ and $\vec{\beta}$ to obtain our results. Qualitative comparisons are provided in Fig. 7, demonstrating that our method achieves performance comparable to [Zhou et al. 2018], with both outperforming the results from [Weng et al. 2013]. Note that in the middle column of interpolation, our method generates strands naturally covering the forehead, rather than severely intersecting with the head mesh like [Weng et al. 2013]. Our starting and ending hairstyles are a bit different from

others, which are caused by the parameterization process, as some details cannot be fully reconstructed.

C.5 Single-view Hair Reconstruction

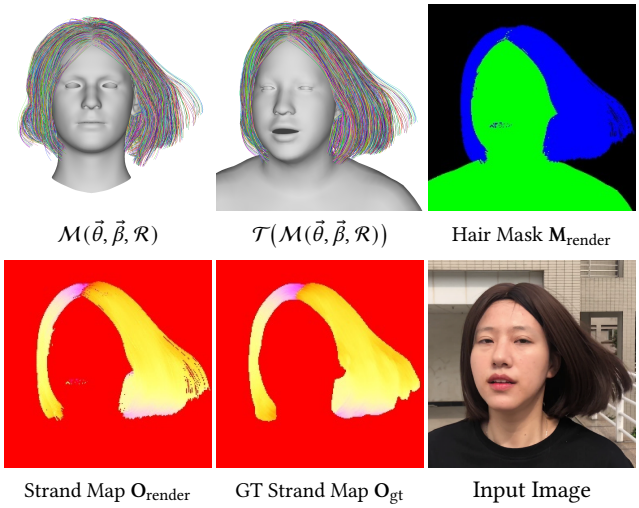


Fig. 8. Illustration of our differentiable rendering pipeline for single-view hair reconstruction.

Given RGB images as input, we first run DELTA [Feng et al. 2023] to estimate the SMPL-X [Pavlakos et al. 2019] face and shoulder geometry with the projecting camera parameters. We then segment hair from the image and compute its strand map as described in HairStep [Zheng et al. 2023], which depicts the 2D pixel-wise hair orientations. To fit 3D hair to 2D orientations, we design a differentiable rendering pipeline, as illustrated in Fig. 8. In the pipeline, we pre-compute the transformation between our in-house head mesh and the SMPL-X model, denoted as the function $\mathcal{T}$, thus placing strand polylines generated from Perm onto SMPL-X. We then attach a cylinder mesh onto each strand segment, as well as computing the 3D orientation of each vertex as the per-vertex feature. These features are projected and rendered using Nvdiffrast [Laine et al. 2020] with the estimated camera parameters, thereby obtaining the rendered hair mask $\mathbf{M}_{\text{render}}$ and strand map $\mathbf{O}_{\text{render}}$. By computing the pixel-wise mask loss and strand map loss similar to HairStep [Zheng et al. 2023], gradients are back-propagated to optimize $\vec{\theta}$ and $\vec{\beta}$ with decoupled weight decay regularization [Loshchilov and Hutter 2017]. A penetration loss is applied as well to penalize hair intersecting the body geometry.

To accommodate the computation in affordable GPU memory, we generate hair with randomly picked sparse roots for each iteration. To initialize $\vec{\theta}$ and $\vec{\beta}$, we search in the original USC-HairSalon dataset for the hair that has the lowest mask and strand map loss, and compute its Perm parameters as described in Sec. C.3.

In Fig. 9 we show our single-view hair reconstruction results on various input images with curly hairstyles and tilted head poses, and compare our results to HairStep [Zheng et al. 2023]. To test reconstruction for hair under dynamics, we run our algorithm on image sequences sampled from a video in [Yang et al. 2019], and compare our results to both [Yang et al. 2019] and HairStep [Zheng et al. 2023] in Fig. 10. Since Yang et al. [2019] have not released the training data or pre-trained model of their method, we can only use their provided videos for comparison. Among these methods, ours method achieves the best reconstruction quality regarding both the global hair shape and local strand details, while also avoiding artifacts such as bald areas observed in HairStep. Despite being trained solely on static data, our model demonstrates the ability to generalize and capture the dynamic effects of hair in the images. We conjecture that it is because our disentangled guide strand parameters are flexible enough to capture large hair deformation variations. Additionally, since our reconstructed hairstyles are represented as parameters, we can substitute their $\vec{\beta}$ parameters with that of a wavy hairstyle (shown as reference in the last row), thereby transferring the wavy style to the reconstructed results while preserving their overall shapes.

C.6 Hair-conditioned Image Generation

In Fig. 11 we present a comparison of image generation results with and without our input hair conditions. Note that we use the previsualization from MeshLab [Cignoni et al. 2008] as the structural image condition, which we found to perform better than the final renderings. Without our hair conditions, images generated with rough text prompts like *"wavy and short hair"* cannot guarantee the production of the desired hairstyle, and their hairstyles often vary with pose, resulting in a loss of multi-view consistency. Leveraging the rich information encoded in the 3D hair geometry, our hair conditions effectively address these issues, producing high-quality portrait images with a more consistent hairstyle. In Fig. 12 we further showcase some generated images conditioned on various input hairstyles. The texture colors in the input hairstyle renderings are only for aesthetic purpose and have no effect on the skin or hair colors in the generated images. Although structural conditioned image generation is not our focus and needs more future investigation on improving the structural alignment, this application reveals the potential of deploying current T2I models as a "neural renderer" [Tewari et al. 2020] within the traditional CG pipeline.

REFERENCES

Adobe. 2024. Firefly. https://www.adobe.com/products/firefly.html.

Jorge Amador Herrera, Yi Zhou, Xin Sun, Zhixin Shu, Chengan He, Soren Pirk, and Dominik L. Michels. 2024. DigitalSalon: A Fast Simulator for Dense Hair via Augmented Mass-Spring Model. *preprint* (2024). https://repository.kaust.edu.sa/items/b4a5000e-58c8-4b94-aa88-0e2638884b4c

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, Vittorio Scarano, Rosario De Chiara, and Ugo Erra (Eds.). The Eurographics Association. https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

Yao Feng, Weiyang Liu, Timo Bolkart, Jinlong Yang, Marc Pollefeys, and Michael J. Black. 2023. Learning Disentangled Avatars with Hybrid 3D Representations. *arXiv* (2023).

Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-Excitation Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7132–7141. https://doi.org/10.1109/CVPR.2018.00745

Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. 2015. Single-View Hair Modeling Using A Hairstyle Database. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2015)* 34, 4 (July 2015).

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and Improving the Image Quality of StyleGAN. In *Proc. CVPR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Fig. 9. Single-view hair reconstruction on images with curly hair and tilted head poses, with comparison to HairStep [Zheng et al. 2023].

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics* 39, 6 (2020).

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

Linjie Luo, Hao Li, and Szymon Rusinkiewicz. 2013. Structure-aware hair capture. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision*. 5442–5451.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. 2019. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10975–10985.

Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. 2021. Encoding in Style: a StyleGAN Encoder for Image-to-Image Translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. 2022. Neural Strands: Learning Hair Geometry and Appearance from Multi-View Images. *ECCV* (2022).

Yuefan Shen, Shunsuke Saito, Ziyan Wang, Olivier Maury, Chenglei Wu, Jessica Hodgins, Youyi Zheng, and Giljoo Nam. 2023. CT2Hair: High-Fidelity 3D Hair Modeling using Computed Tomography. *ACM Transactions on Graphics* 42, 4 (2023), 1–13.

Vanessa Sklyarova, Jenya Chelishev, Andreea Dogaru, Igor Medvedev, Victor Lempitsky, and Egor Zakharov. 2023. Neural Haircut: Prior-Guided Strand-Based Hair Reconstruction. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. 2020. State of the art on neural rendering. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 701–727.

Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. 2009. Example-Based Hair Geometry Synthesis. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 56, 9 pages. https://doi.org/10.1145/1576246.1531362

Yanlin Weng, Lvdi Wang, Xiao Li, Menglei Chai, and Kun Zhou. 2013. Hair interpolation for portrait morphing. In *Computer Graphics Forum*, Vol. 32. 79–84.

Lingchen Yang, Zefeng Shi, Youyi Zheng, and Kun Zhou. 2019. Dynamic hair modeling from monocular videos using deep neural networks. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.

Yujian Zheng, Zirong Jin, Moran Li, Haibin Huang, Chongyang Ma, Shuguang Cui, and Xiaoguang Han. 2023. HairStep: Transfer Synthetic to Real Using Strand and Depth Maps for Single-View 3D Hair Modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yuxiao Zhou, Menglei Chai, Alessandro Pepe, Markus Gross, and Thabo Beeler. 2023. GroomGen: A High-Quality Generative Hair Model Using Hierarchical Latent Representations. *arXiv preprint arXiv:2311.02062* (2023).

Yi Zhou, Liwen Hu, Jun Xing, Weikai Chen, Han-Wei Kung, Xin Tong, and Hao Li. 2018. Hairnet: Single-view hair reconstruction using convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 235–251.

Fig. 10. Single-view hair reconstruction and editing on image sequences, with comparison to [Yang et al. 2019] and HairStep [Zheng et al. 2023].

Fig. 11. Comparison of image generation with and without hair conditions. These images are generated with the same text prompt *"wavy and short hair, white sweater"*. Additional text prompts like *"front face"* or *"side face"* are appended to assist the head pose in the generated images.

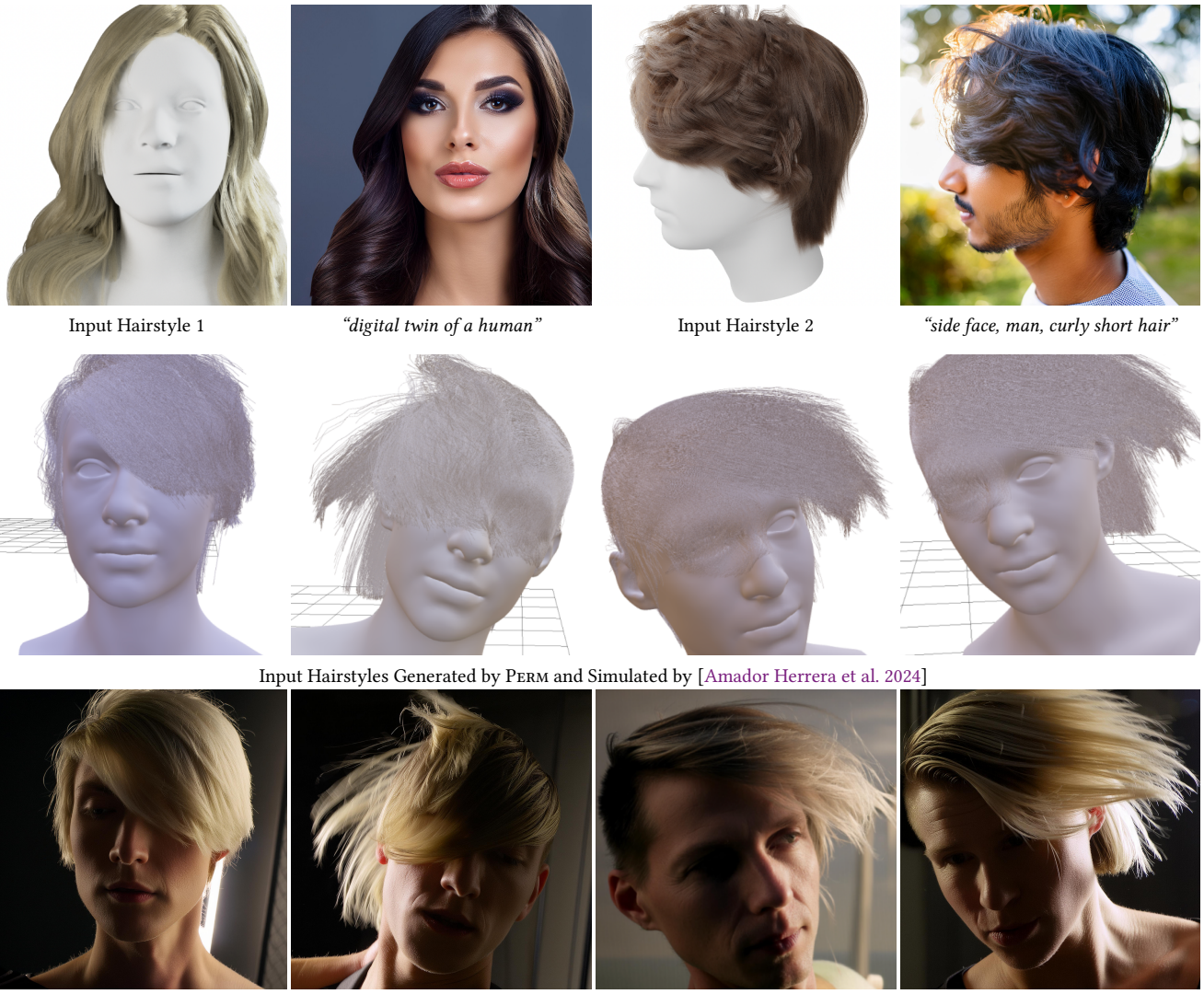


Fig. 12. Hair-conditioned image generation using Adobe Firefly [Adobe 2024].