

Quaffure: Real-Time Quasi-Static Neural Hair Simulation

Supplementary Material

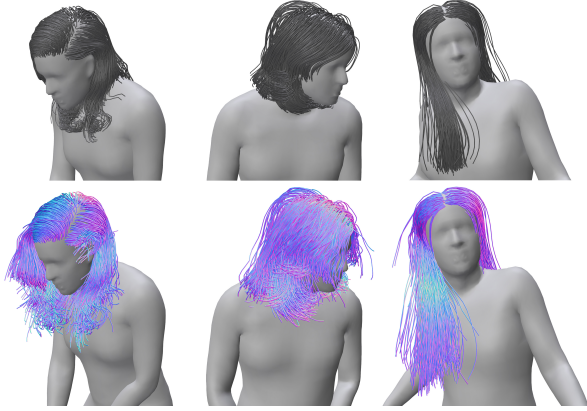


Figure 1. We apply our trained groom deformation decoder on novel grooms, unseen during training. Top row shows the groom transformation module output and bottom row shows the full predicted result. Note that predicted results are still reasonable for grooms that are similar to those in the training set.

1. Groom Generalization

We train each of our networks on 10 randomly selected grooms and in the main paper, we showcase how the method generalizes to body pose and shape variations. Although efficiency was our design goal and not groom generalization, we investigate the results of decoding groom deformations for unseen grooms which were not used for training. See Figure 1 for visual results. As expected, prediction quality is degraded when compared to the training set. However, note how the method produces reasonable deformations where the strands follow the gravity direction, we also note that deformations are more limited and muted, staying closer to the result of the groom transformation module. Given our real-time goals, we opted for a small model size. Despite that, our model can already handle at least 10 distinct grooms. We acknowledge that a larger decoder may be needed to scale to more grooms, sacrificing inference performance as discussed in the main paper.

2. Network Structures

In this section we provide more details about our network architectures, namely the groom autoencoder, which is used for training the groom latent space, and the groom deformation decoder, which takes as input the groom latent code, body shape and body pose, and predicts strand motion relative to the rigid body motion.

2.1. Groom Autoencoder

In order to build a latent groom space, we train a groom autoencoder. We define the hair groom as a texture of dimension $T \times T \times 3N$ on the head scalp, uniformly distributing strands across the scalp. In all experiments we use texture size of $T = 64$ and number of vertices per strand $N = 24$. An encoder takes the 2D groom representation (with channel size of $3N$) as input, and encodes it using a series of convolutional layers, decreasing the spatial dimension to 4×4 while gradually increasing channel size ($128 \rightarrow 256 \rightarrow 512 \rightarrow 1024$). Each layer consists of a residual block and convolutional downsampling. Finally, a linear layer is applied to convert feature texture into a latent code of dimension 16. Groom decoder follows an equivalent convolutional architecture, but in reverse. Upsampling at each layer is done using transpose convolutions. As activations at each layer, we use a sigmoid linear unit (SiLU) function.

2.2. Groom Deformation Decoder

To predict groom deformation given body shape and pose parameters, we train a groom deformation decoder. The input to the model consists of a groom latent code (of dimension 16), body shape (of dimension 10) and pose (of dimension 81) parameters, and the output is a texture of dimension $T \times T \times 3N$, where the groom is again defined on the head scalp (with $T = 64$ and $N = 24$). However, in the case of deformation decoder, we predict only the position difference to the rigidly moved positions, instead of absolute positions. The decoder architecture is similar to the groom autoencoder, starting with a linear layer and continuing with convolutional upsampling (adapting channel size as $1024 \rightarrow 2048 \rightarrow 2048 \rightarrow 1024$).

3. Implementation and Training Details

We leverage a vast data set of motion capture data which includes a large variety of motions. For every motion recording, we randomly sample poses from the sequence. In addition, to model shape variations, we randomly sample body shape coefficients at training time. In all of our results, strands consist of $N = 24$ vertices which are encoded into 64×64 texture maps. This choice of texture dimensions provides us with the ability to model several thousands of hair strands which is sufficient for guide hair simulation. Prior work [2] shows high quality upsampling results for as little as 128 guide strands. For better visual results, we rigidly fix the first M to be fully constrained to the output of the groom transformation module, we do this to main-

tain the intended hair style and strand orientations near the root. In our results, $M = 8$. Our method does not require parameter tuning for different hairstyles and all results are obtained using the following values: $k_{\text{stretch}} = 10000$, $k_{\text{cosserat}} = 3000$, $k_{\text{bc}} = 5000$, $k_{\text{sc}} = 500$, $k_{\text{pr}} = 10$, $h = 0.5$ and $D = 1.0$. All results are obtained using the Adam optimizer [3] with a learning rate of $1.0e^{-5}$. We train using 10 grooms and 16 poses per batch where $N_{\text{pose.reg}} = 4$. We implement our method with PyTorch [4], with training and inference measurements performed on an AMD Ryzen Threadripper PRO 3975WX CPU and a single NVIDIA RTX A6000 GPU. We make use of Polyscope [5] for visualization and Blender [1] for producing path-traced renders.

References

- [1] Blender. Blender. <https://www.blender.org//>, 2024. Accessed on November 2024. 2
- [2] Jerry Hsu, Tongtong Wang, Zherong Pan, Xifeng Gao, Cem Yuksel, and Kui Wu. Real-time physically guided hair interpolation. *ACM Transactions on Graphics (TOG)*, 43(4):1–11, 2024. 1
- [3] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2
- [4] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017. 2
- [5] Nicholas Sharp et al. Polyscope, 2019. www.polyscope.run. 2