Real-Time Hair Simulation with Neural Interpolation

Qing Lyu, Menglei Chai, Xiang Chen†, and Kun Zhou, *Fellow, IEEE*

Abstract—Traditionally, reduced hair simulation methods are either restricted to heuristic approximations or bound to specific hairstyles. We introduce the first CNN-integrated framework for simulating various hairstyles. The approach produces visually realistic hairs with an interactive speed. To address the technical challenges, our hair simulation pipeline is designed as a two-stage process. First, we present a fully-convolutional neural interpolator as the backbone generator to compute dynamic weights for guide hair interpolation. Then, we adopt a second generator to produce fine-scale displacements to enhance the hair details. We train the neural interpolator with a dedicated loss function and the displacement generator with an adversarial discriminator. Experimental results demonstrate that our method is effective, efficient, and superior to the state-of-the-art on a wide variety of hairstyles. We further propose a performance-driven digital avatar system and an interactive hairstyle editing tool to illustrate the practical applications.

Index Terms—real-time hair simulation, neural interpolator, generative models, computer animation, CNN, GAN

1 Introduction

Hairstyle is an essential ingredient of fashion that reflects the individual personality of human beings in real life. With the increasing popularity of AR/VR applications, digitizing the hairs becomes a vital job of 3D avatar modeling. Empowered by recent advances in 3D synthesis techniques, a wide variety of hairstyles can be faithfully captured from single images [1], [2]. Nonetheless, the social nature of the digital avatars still requires these hair models to be responsive to all kinds of user interactions.

Endless efforts have been made to pursue the realism of hair animation, ranging from non-smooth frictions [3], impact and collisions [4], to complex hair-liquid interactions [5]. Although these methods can generate highly realistic details, they do not directly apply to interactive applications due to the cost of computations. Traditional hair simulation methods for real-time usages usually simplify the intricate hair-hair interactions through heuristic models such as clumped structures [6], [7] and continuum formulations [8], [9], which often lead to deteriorated hair details. Recently, Chai *et al.* [10], [11] present a data-driven method to reduce the complexity of hair-hair interactions by the knowledge extracted from full simulation data. It physically simulates a small set of guide hairs and interpolates all other hair strands from them. The method succeeds in producing visually realistic hairs at interactive speed. However, it first determines the guide hairs by graph partitioning and then fixes the set of weights for interpolation, which binds the reduced model to a specific hairstyle and prohibits its generalization to unseen data. A cumbersome and expensive retraining process is thus unavoidable for any novel hairstyle.

To address the above challenges, we introduce a neural interpolator to compute the weights *on-the-fly* and interpolate the guide hairs accordingly. We predict these dynamic weights by taking the rest shape of the hairstyle and the run-time states of the guide hairs as input. By training on the data from full simulations, we extract the knowledge of intricate hair-hair interactions from the simulation data and embed it into a fully convolutional neural network. The learned parameters of the resultant CNN form a *unified* representation space for all the training data, which naturally generalizes to unseen hairstyles due to its inherent localness. This neural interpolator plays the role of an expressive and robust backbone in our hair simulation framework. To further enhance the realism of hair details, we train another light-weighted generator based on the generative adversarial networks (GAN) architecture. The fine-scale displacements produced by this generator is superimposed onto the backbone's output to obtain the final results.

We demonstrate both the efficacy and efficiency of our method by simulating a wide variety of hairstyles at interactive speed. Experiments show that our neural interpolator produces much more perceptually realistic results than the state-of-the-art methods. Furthermore, we demonstrate the practical applications of our approach with a performance-driven digital avatar system and an interactive hairstyle editing tool.

Our main contributions are:

- The first CNN-integrated framework for simulating various hairstyles. Compared with the expensive full simulations, our method produces perceptually similar results with an interactive speed.
- A neural interpolator that predicts the dynamic weights for hair interpolation, which enables both the expressive representation and the robust generalization.
- An adversarially trained displacement generator that produces fine-scale hair details enhancing the visual

- *Q. Lyu, X. Chen and K. Zhou are with the State Key Lab of CAD&CG, Zhejiang University.*
- *M. Chai is with Snap Research.*

†*Corresponding author. E-mail: xchen.cs@gmail.com*

realism.

2 Related Work

2.1 Physics-Based Hair Simulation

Realistic hair simulation has been an active area of computer graphics research in the last decades [12], [13]. A compact yet expressive mechanical model of individual strands is at the core of high-fidelity hair dynamics. Representative studies include mass-spring models [14], [15], [16], projective dynamics [17], [18], rigid body chains [7], [8], and Kirchhoff inextensible elastic rods [19], [20]. Recently, researchers have extensively explored the complex hair-hair interactions such as stiction [16], self-collision [21], Coulomb friction [3], [22], and transversal impact [4] to enhance physical realism persistently. In this work, we rely on these advances to efficiently produce high-fidelity simulations for a small set of representative hairs. Specifically, the generalized altitude spring model [16] is adopted due to its efficiency and versatility, though we can substitute it with other alternatives.

2.2 Reduced Hair Simulation

Previous works put much effort into the acceleration of complex hair simulation. For example, the computation of hair-hair interactions is often simplified via continuum strategies like smoothed particle hydrodynamics [8], [9], regular grid [23], [24], hybrid Eulerian/Lagrangian approach [21], and hair meshes [25], [26]. The methods most relevant to this work simulate a small set of clumped wisps at the coarse-level and generate the individual strands with heuristic or statistical models [6], [7], [15], [19], [27], at the cost of degrading or over-smoothing the hair details.

Chai *et al.* present a reduced data-driven method [10] to extract a small set of guide hairs from full simulations and compute the corresponding weights for interpolating other normal hairs. In [11], they further extend the method to handle hair-solid simulation by adaptively choosing the best group from multiple sets of guide hairs at runtime and bilaterally resolving the collisions. In their work, the finite sets of guide hairs and the interpolation weights are both precomputed on a specific simulation dataset. In contrast, we do not explicitly compute and store the interpolation weights. We instead learn a fully-convolutional network that predicts the interpolation weights on-the-fly according to the current states of the guide hairs. The dynamically computed weights are thus *states-aware* and enable a simulation result with much more realistic details. More importantly, the weight pre-computation in previous works is global and bound to a specific hairstyle, which prohibits its use on a heterogeneous dataset with many hairstyles. On the contrary, we extract knowledge from representative hairstyles and embed the knowledge into the network by leveraging its inherent localness.

2.3 Neural Network Based Methods

Recently, neural networks and relevant techniques [28] have seen various applications in computer graphics. Here we only review the most relevant ones to our study. Bailey *et al.* [29] propose a fully-connected network for fast correction of the skinning result to approximate the intricate and realistic shape deformation effects of general character meshes. Similarly, our fine-scale displacement is also a correction strategy specific for hair details. Liu *et al.* [30] design a graph-convolutional neural network based on the attention mechanism to automatically bind the shape of an arbitrary character to a given articulation skeleton, whose binding weights are computed for skinning. Our method can be seen as a solution to the adaptive per-frame binding problem between the normal hairs and the guide hairs in their current states. Traditionally, fast simulation of elastic bodies leverages a manually-designed warping step to approximate the time-variant stiffness matrix computation. In contrast, Luo *et al.* [31] adopt a fully-connected neural network to learn a warping function from the precomputed simulation data. Their method enables generalization to various shapes by predicting per-point warping, while our method achieves generalization via the localness of a convolutional architecture. Fulton *et al.* [32] employ an autoencoder network to extract an expressive nonlinear subspace for deformable object simulation. Differently, our neural network produces coefficients of a nonlinear subspace spanned by guide hair simulations. Xie *et al.* [33] introduce the techniques of 2D image super-resolution [34] into the fluid simulation domain. A convolutional network is pre-trained adversarially to generate a visually realistic high-resolution result consistent with the low-resolution smoke simulation. In this work, we aim for the real-time simulation problem of hair strands with various styles, where we regard the normal hairs as a super-resolution state of the guide hairs at the current timestep. We also adopt a GAN-like formulation [35] to train a secondary generator but only use it to generate fine-scale displacements to strengthen the hair details.

Prevalent neural network architectures like VAE [36] and GAN have been tailored for hair modeling [1], [2], [37]. Recently, Zhou *et al.* [38] incorporate a collision loss into the training process to penalize the interpenetrations between the hairs and the head, while we design a snapping loss term dedicated to guide-hairs interpolation to prevent such artifacts. Lombardi *et al.* [39] present a VAE-based volume representation for novel view synthesis, where a low-res linear blend of affine warps is learned to mitigate the memory usage. We share the opinion that the skinning formulation can be leveraged to avoid overfitting and enable an effective generalization. With guide strokes as the input, Olszewski *et al.* [40] introduce a GAN-based architecture to synthesize and edit 2D images of facial hairs directly. All the above work targets the hair modeling and synthesis from single-view or multi-view images, while we learn from 3D data to reconstruct high-resolution hair dynamics from low-resolution physics-based simulations.

3 Overview

A visually realistic hair model has more than dozens of thousands of hair strands. Simulating such a large model in real-time is challenging. We follow the spirit of data-driven hair simulation [10] to achieve high-quality hair dynamics in a limited time budget. In the reduced hair simulation method, the hair dynamics are implicitly represented as the interpolation relationships between sparsely sampled

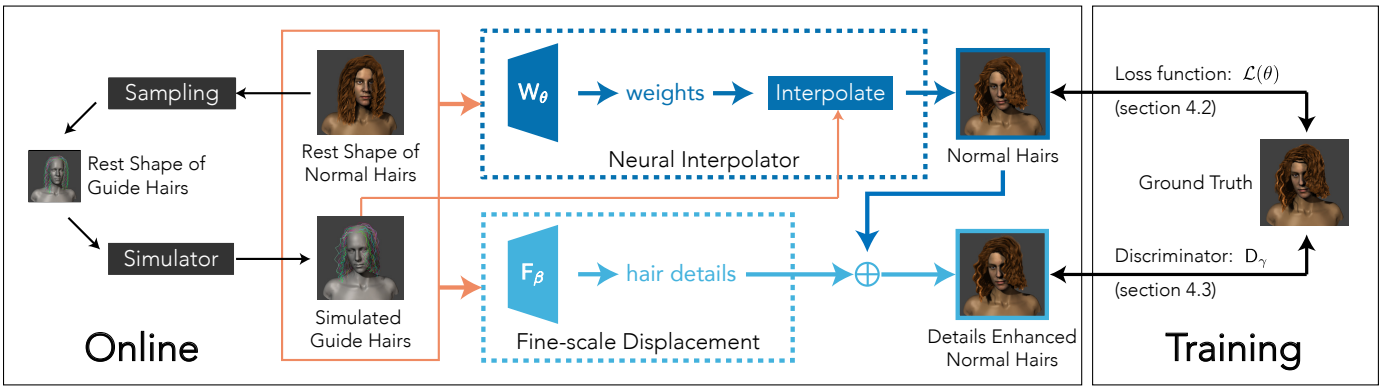


Fig. 1: **Hair Simulation Process.** Our method takes the rest shape, and the uniformly sampled guide hairs as input. At each timestep, the low-resolution guide hairs are physically simulated with a mass-spring model first. Then the neural interpolator receives the updated guide hair positions and predicts the interpolation weights accordingly. After that, the high-resolution normal hairs are interpolated by the guide hairs in their vicinity using the predicted weights. Finally, the fine-scale displacements are generated to enhance the visual realism of the hair details. The neural interpolator is trained with a dedicated loss function, and the fine-scale displacement generator is trained in a GAN architecture.

guide hairs and the normal hairs in their vicinity. However, this kind of relationship is only stable under a specific hairstyle with restricted dynamics and does not generalize to the case of multiple hairstyles. Different hair models are diverse at shapes, styles, and physical properties, which leads to distinctive inter-relationships among adjacent hair strands. This fact prohibits the direct transferring of the global dynamics from one hair model to another. Ideally, a data-driven simulator should be able to capture the local properties and dynamics from representative hairstyles and generalizes them to an unseen hair model. Reflecting such considerations, we design our hair simulation pipeline as a two-stage process, which includes a neural interpolator that computes dynamic weights to interpolate the guide hairs, and a fine-scale displacement generator that further enhances the motion details (Figure 1).

3.1 Online Simulation

The online stage starts with the physics-based simulation [16] of the *guide hairs* to generate representative motions and resolve collisions in the coarse resolution. These guide hairs are sparsely selected and thus fast to simulate. The simulation states of the guide hairs are then pushed into a neural interpolator (Section 4.2.1) to predict the interpolation weights for each normal hair strand of the full resolution model. This U-net-like network produces weights for each normal hair based on the geometry of the rest normal hairs and the current guide hairs, which is pre-trained on the full simulation data of many hairstyles. After that, the states of all the *normal hairs* are computed in the sense of linear blend skinning by mixing the guide hair states according to the predicted interpolation weights. Finally, a second light-weighted neural network (Section 4.3.1), adopted for hairs with extremely high resolution, is executed to compute the fine-scale displacements for all the normal hairs to promote visual realism. The above process is repeated for each timestep during the simulation.

3.2 Training

We train the networks on several time sequences of full simulation data. We simulate multiple hairstyles using a mass-spring model [16] driven by random head motions. For different hairstyles, we choose different dynamic parameters to generate visually realistic movements. Thanks to the parameterization strategy (Section 4.1) and the neural interpolator, the heterogeneous simulation data can be trained simultaneously. The online simulation uses the same set of dynamic parameters as its offline counterpart, while the held-out head motions have no intersections with the training data. The training of the neural interpolator is supervised under a dedicated loss function (Section 4.2.2), and the fine-scale displacement generator is adversarially trained with a discriminator (Section 4.3.2).

4 Method

We now present the key designs and algorithms that enable the learning and fast simulation of multiple hairstyles.

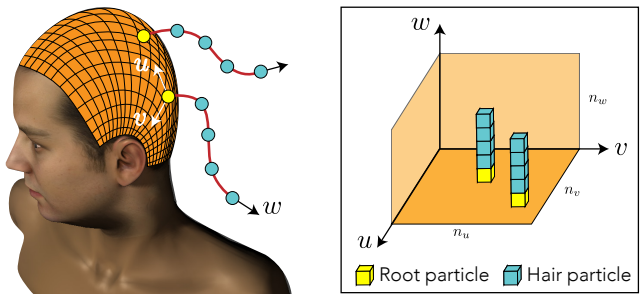


Fig. 2: **Hair State Parameterization.** Left: The discrete hair particles, i.e., the regular grid points embedded in the 3D Euclidean space. Right: The regular grid in the parametric space. Please see Appendix A for the details of the scalp parameterization.

4.1 Parameterization

Learning from the complex and heterogeneous hair dynamics requires a simple yet effective representation of various hairstyles. Here we present a unified parameterization for distinct hair models. First, we parameterize the root particles of all the hair strands as a $n_u \times n_v$ regular-grid on the scalp surface (Figure 2). Second, each hair strand has n_w particles uniformly sampled by length. In this way, we represent the hair model with an arbitrary state as a function defined on a grid of size $n_u \times n_v \times n_w$:

$$s : \Omega \subset \mathbb{N}^3 \rightarrow \mathbb{R}^{n_c}. \tag{1}$$

Therefore, a hair state in this discrete setting is just a fourth-order tensor $\mathbf{s}$ stacking all the state values at the grid points, which is readily available for computation on a neural network. The tensor $\mathbf{s}$ has a size of $[n_u, n_v, n_w, n_c]$, where n_c, the state channels of each hair particle, is set to 3 since we only store the Euclidean coordinates [1]. In what follows, we choose the values $n_u = n_v = 128, n_w = 25$ if they are not explicitly specified. Besides the state tensor $\mathbf{s}$, we also need the rest state of each hairstyle. We denote it as a tensor $\bar{\mathbf{s}}$ with the same size.

The state tensor $\mathbf{s}$ encodes the inter-hairs and intra-hair neighborhood information into its cartesian index, which naturally enables a convolutional network to extract multi-levels of local features from the hairs.

In the u and v directions, we uniformly select $n_g \times n_g$ strands as the representative guide hairs (the dots in Figure 3), whose state tensor $\mathbf{g}$ with a size of $[n_g, n_g, n_w, n_c]$ are updated by a physics-based simulator at runtime. In this paper, we set $n_g = 16$ for all the experiments.

Note that the hair states involved in the neural network computations below are in the head's local space. We obtain the hair states in the world space by multiplying the local states with the rigid transformation of the head.

4.2 Neural Interpolator

By leveraging the tensor-based representation of hair states, we present a neural hair state interpolator by taking advantage of recent progress in machine learning research.

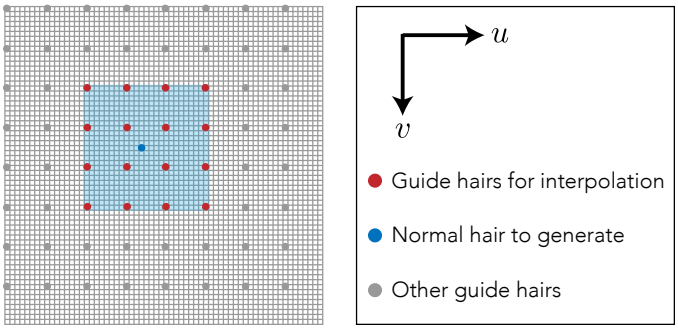


Fig. 3: **Guide Hair Interpolation.** The uniform sampling of the guide hairs in the uv plane. A normal hair particle is interpolated from its nearby 16 guide hair particles.

1. We haven't observed remarkable promotion by adding three extra channels for the particle velocity, which is similarly mentioned in [33].

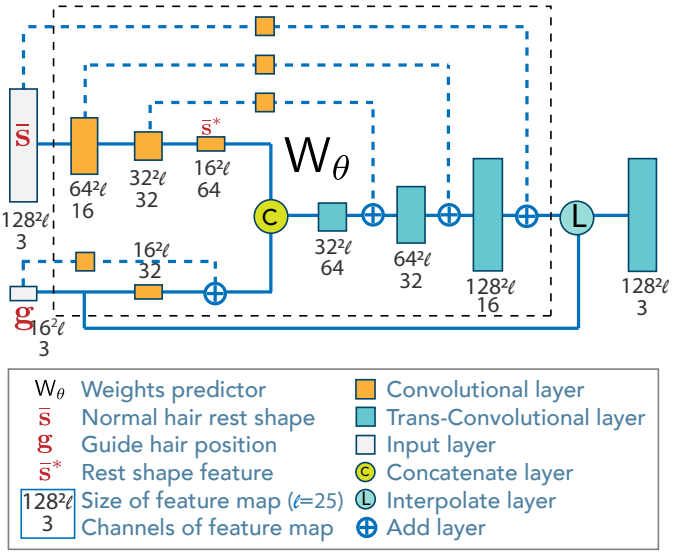


Fig. 4: **Neural Interpolator Architecture.** The neural interpolator receives the rest shape and the current guide hair positions, to generate the interpolation weights. A sequence of convolutions and transposed convolutions are skip connected like the U-net [41] architecture. The high-resolution normal hairs are interpolated from the guide hairs in their vicinity using the predicted weights.

4.2.1 Architecture

The neural interpolator consists of two components, i.e., the weights predictor W_θ with trainable parameters θ and the linear interpolator L (Figure 1). The weight predictor takes the rest state $\bar{\mathbf{s}}$ and the guide hair states $\mathbf{g}$ updated in current timestep as input, and produces the interpolation weights. Then the states of the guide hairs are linearly interpolated with these predicted weights to update the states $\mathbf{s}$ of the normal hairs:

$$\mathbf{s} = \mathsf{L}(\mathbf{g}, \omega), \tag{2}$$

where $\omega = \mathsf{W}_\theta(\bar{\mathbf{s}}, \mathbf{g})$ is the weight tensor with a size of $[n_u, n_v, n_w, n_n]$. During the interpolation, for each normal hair strand, we take only n_n guide hairs in its closest square neighborhood into consideration (the red dots in Figure 3). That means, for each particle $\mathbf{p}$ on a normal hair strand, W_θ produces a weight vector $\omega_\mathbf{p}$ of length n_n for its interpolation. We use $n_n = 16$ in all our experiments. Then the state $\mathbf{s}_\mathbf{p}$ of this particle is updated as:

$$\mathbf{s}_\mathbf{p} = \sum_{\mathbf{q} \in \mathcal{G}_\mathbf{p}} \mathbf{g}_\mathbf{q} \cdot \omega_{\mathbf{pq}}. \tag{3}$$

Here $\mathcal{G}_\mathbf{p} = \mathsf{adj}(\mathbf{p})$ denotes the index set of guide hair particles in the neighborhood of $\mathbf{p}$, and thus we have $|\mathcal{G}_\mathbf{p}| = n_n$. For any boundary hair $\mathbf{p}$ with its neighborhood not fully covered by the uv map, we simply pad $\mathcal{G}_\mathbf{p}$ with the nearest valid guide hairs to ensure proper interpolation.

The weights predictor network W_θ is composed of a set of convolution and transposed convolution layers correlated by skip connections (see Figure 4). Specifically, we generate a pyramid of rest shape features by a sequence of convolutions, and concatenate the coarsest one $\bar{\mathbf{s}}^*$ with the features

extracted from current guide hairs. The concatenated features then walks through a sequence of transposed convolutions to get back to the original resolution. LeakyReLU is used as the activation function for the hidden layers, together with instance normalization. A softmax is applied at the end of the network to normalize the interpolation weights to satisfy the convex constraints, i.e. $\omega_{\mathbf{pq}} \geq 0$ and $\sum_{\mathbf{q}} \omega_{\mathbf{pq}} = 1$. This architecture is fully-convolutional, which ensures that the features are locally extracted from adjacent hair strands by a series of spatially sharable kernels. Relied on this locality, we can apply the trained network to various hairstyles with distinct spatial arrangements.

4.2.2 Loss Function

During the offline training phase, we compute the convolutional network parameters θ by minimizing a loss function dedicated to hair simulation:

$$\mathcal{L}(\theta) = 0.01\mathcal{L}_{data} + 1\mathcal{L}_{len} + 0.1\mathcal{L}_{snap} + 0.1\mathcal{L}_{tempo}, \quad (4)$$

where the loss function $\mathcal{L}$ is composed of four loss terms, i.e., the primary data regression term $\mathcal{L}_{data}$, the hair strand length-preservation term $\mathcal{L}_{len}$, the guide hair mode snapping term $\mathcal{L}_{snap}$, and the temporal coherence term $\mathcal{L}_{tempo}$. We empirically choose the weights given in Equation 4 to balance the non-homogeneous scales of these terms.

For each frame, we penalize the L1 norm of the deviation between the ground-truth states $\hat{\mathbf{s}}$ and the normal hair states $\mathbf{s}$ generated by the neural interpolator (Equation 2), which ensures a sufficiently close data reconstruction:

$$\mathcal{L}_{data} = \left\| \hat{\mathbf{s}} - \mathbf{s} \right\|_1 . \quad (5)$$

Since a hair strand is physically easy to bend but almost non-stretchable, we penalize the length variation of every normal hair segment with respect to the rest state:

$$\mathcal{L}_{len} = \sum_{\mathbf{a},\mathbf{b} \in \text{segments}} \left(\|\mathbf{s_a} - \mathbf{s_b}\|_2 - \|\bar{\mathbf{s}}_\mathbf{a} - \bar{\mathbf{s}}_\mathbf{b}\|_2 \right)^2 , \quad (6)$$

where $\mathbf{a}$ and $\mathbf{b}$ are the start and the end points of a hair segment. Though $\mathcal{L}_{data}$ implicitly encodes the inextensibility condition, we explicitly factorize the violation of length from the data term to better control the regularization.

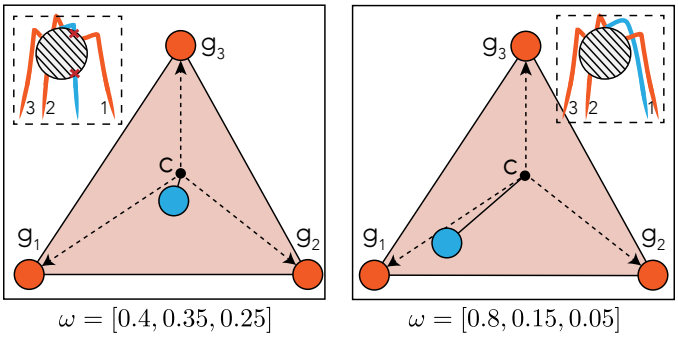


Fig. 5: **Snapping to One Guide Hair Mode.** Naive interpolation weights lead to interpenetration artifacts (left). Meaningful weights produced by our method generate reasonable interpolation results (right).

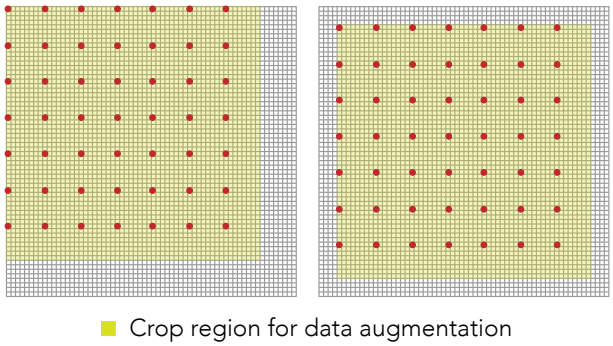


Fig. 6: **Data Augmentation.** Two valid crops (yellow) augmenting the training data. We totally generate 64 valid crops per-frame.

The guide hair states keep changing during the motion. For each normal hair strand, the adjacent guide hairs responsible for its interpolation may converge together or spread out according to their mutual interactions. When this subset of guide hairs has several completely different modes, naive interpolation among these quasi-discrete modes is meaningless, which quickly leads to physically incorrect results such as interpenetration (left of Figure 5). Considering such facts, we intentionally *snap* the normal hair to one particular mode of its adjacent guide hairs by penalizing the vanishing sum of the centrifugal directions:

$$\mathcal{L}_{snap} = -\sum_{\mathbf{p}} \left\| \mathbf{s_p} - \frac{1}{|\mathcal{G}_\mathbf{p}|} \sum_{\mathbf{q} \in \mathcal{G}_\mathbf{p}} \mathbf{g_q} \right\|_2 , \quad (7)$$

which vanishes, i.e., the distinct modes cancel out each other, when the predicted weights interpolate the guide hairs in an almost uniform way (that we want to avoid). Incorporating the loss term, the trained network produces reasonable weights that interpolate guide hairs mainly from one particular mode (right of Figure 5).

The loss terms we have introduced so far are imposed at each frame independently. Flickering artifacts would appear due to the possible violation of the temporal coherence (see the video and Figure 13). Therefore, we incorporate a loss term to penalize the inconsistencies in the temporal space:

$$\mathcal{L}_{tempo} = \sum_{t} \left\| \left(\mathbf{s}^{(t+1)} - \mathbf{s}^{(t)} \right) - \left(\hat{\mathbf{s}}^{(t+1)} - \hat{\mathbf{s}}^{(t)} \right) \right\|_1 , \quad (8)$$

where we use the forward difference formula to compute the predicted and ground-truth velocity at frame t.

4.2.3 Data Augmentation

Inherently, the neural interpolator captures the knowledge of mutual relationships among guide hairs and uses that to predict the corresponding normal hairs. Therefore, extracting only a single set of guide hairs for each hair frame is inefficient for the training. Instead of using a fixed pattern for selecting guide hairs, we further augment the training data by sliding a crop window in the uv plane (see Figure 6). Specifically, we make every slot of the 8×8 (for 128 resolution) upper-left region in the uv plane as the corner of a valid crop, which gives us 64 crops per-frame. We regard each crop as a ground-truth data $\hat{\mathbf{s}}$, and again we select one

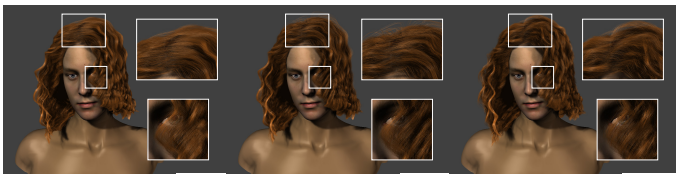


Fig. 7: **Displacement Generator Comparison.** Compared with the ground-truth data (right), the hair details produced by using the fine-scale generator (middle) is visually more realistic than that of without using it (left).

for every eight strands in u and v directions as the guide hairs $\mathbf{g}$. In this way, every hair strand in the frame gets a chance to play the role of a guide hair, which significantly enriches the mutual relationships among the hairs in the training data.

4.3 Fine-Scale Displacement

The neural interpolator described above produces pleasing results for hairs with moderate resolution. However, when the resolution of the normal hairs goes up to 128×128, the hair details gradually deteriorate since we keep using the 16×16 guide hairs. In such a case, the supporting region of each guide hair strand expands larger, which inevitably leads to a smoother interpolation of the normal hairs (see Figure 7). Therefore, we introduce a fine-scale displacement generator to strengthen the realistic details of the neural interpolator's output. Visually, this generator produces a considerable amount of hair details only when the resolution is higher than 128×128, according to our observation (See the supplemental for more comparisons).

4.3.1 Generator

The generator F_β generates a fine-scale displacement:

$$\Delta\mathbf{s} = \mathsf{F}_\beta(\bar{\mathbf{s}}^*, \mathbf{g}), \tag{9}$$

which is added to the neural predictor's output to obtain more realistic hair details $\tilde{\mathbf{s}} = \mathbf{s} + \Delta\mathbf{s}$. Thanks to the efficacy and stability of our neural interpolator introduced above, the generator F_β is made lightweight and consists of only one convolution and two transposed convolutions (see Figure 8), which are sufficient to produce the level of hair details that we need.

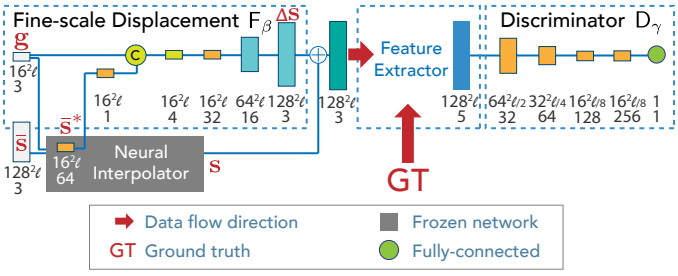


Fig. 8: **Displacement Generator Architecture.** The secondary generator takes as input the current guide hair states $\mathbf{g}$, and the semantic features $\bar{\mathbf{s}}^*$ (extracted from the rest shape). It produces the fine-scale displacement and adds that onto the neural interpolator's output. The result is put into the discriminator for adversarial training.

Fig. 9: **Exemplars of training hairstyles.** Our training data includes multiple hairstyles with different length and curliness.

4.3.2 Discriminator

The adversarial training of the fine-scale displacement generator fails to produce realistic hair details if we directly use the 3D Euclidean coordinates as the input to the discriminator. Instead, we first push the 3D coordinates into a predefined feature descriptor E and then feed its output into the discriminator D_γ for classification. The feature descriptor have five channels per each hair particle, of which the first four channels are the distances to its four closest guide hair particles, and the last channel is the geodesic distance to the root particle on the strand. Empirically, we find the explicit use of such features not only helps capture the semantic details of representative hairstyles but also stabilizes the GAN training. The discriminator is composed of a series of convolutions and a fully-connected layer (see Figure 8). We adversarially train the generator F_β to discriminate between the generated results and the ground-truth:

$$\min_\beta \max_\gamma \left(\log \mathsf{D}_\gamma(\mathsf{E}(\hat{\mathbf{s}})) + \log(1 - \mathsf{D}_\gamma(\mathsf{E}(\tilde{\mathbf{s}})))\right), \tag{10}$$

by alternating the gradient descent update of the network parameters β and γ. In practice, we randomly select a frame data pair $(\hat{\mathbf{s}}, \tilde{\mathbf{s}})$ during each iteration, and update β once and γ twice accordingly in the style of Wasserstein GAN [42].

5 Experimental Results

5.1 Hardware and Software

We train our neural interpolator and fine-scale displacement generator on a workstation with four NVIDIA GTX 1080Ti GPUs. For physics-based simulation and online network inference, we use a desktop PC with an Intel Core i7-4790 CPU and an NVIDIA GTX 1080 GPU. We implement the neural network modules in Keras [43] with TensorFlow backend [44] and the online mass-spring based hair simulation system in C++ with parallel computing module.

5.2 Data Generation and Training

We use ten different hairstyles to prepare the *training data*, in which five are straight hairs, and five are curly hairs, all with different lengths (see the exemplars in Figure 9). We

Fig. 10: Our method generates visually realistic animations for a wide variety of hairstyles in an interactive speed.

drive the physics-based simulation of each hairstyle by an underlying head motion sequence consists of $12,000$ frames. The sequence is precomputed by randomly sampling several key poses and SLERPing the in-between frames with a smoothing constraint of the speed. We produce the *test data* by running the same physics-based simulator as used for training under the same sets of parameters. The hairstyles and head motions have no intersections with the training data. The hairstyles vary in length, curliness, growth direction, and parting style, which are either collected from a public hair dataset [2] or procedurally generated by ourselves.

For the neural network training, we employ the ADAM optimizer [45] with a learning rate of 0.001 and a batch size of 4. We terminate the training process after four epochs for the neural interpolator and one epoch for the fine-scale displacement generator, respectively. See Appendix B and the supplemental for more details about data and architecture.

5.3 Generalization

Based on the hair parameterization and the network design presented in Section 4, we succeed in training a unified yet expressive neural interpolator on representative hairstyles. Figure 10 shows typical hair simulations generated by the resultant neural interpolator. Though the test hairstyles and head motions are distinct from the training data, the simulations preserve sufficient fidelity and are perceptively indistinguishable from the ground-truth. Table 1 shows a list of quantitative comparisons. Rather than directly computing any similarity metrics on 3D hair, we adopt the PSNR index to compare the rendered hair images to approximate human perception.

Hairstyles are innumerable, and the 3D hairs data made public is limited. Therefore, it's neither practical nor necessary to achieve generalization by training a vast network. Fortunately, like the natural images, hairs also possess spatially invariant local features. Hence a good generalization ability heavily relies on the network's local adaptivity. Our neural interpolator is fully-convolutional by design and is

TABLE 1: **Quantitative Comparison.** We execute the comparisons on a single hairstyle to show the superiority of our method.

Method		PSNR
Our method	$\mathcal{L}_{data}$ only	28.92
	w/o $\mathcal{L}_{len}$	28.04
	w/o $\mathcal{L}_{snap}$	28.97
	w/o fine-scale displacement F_{β}	29.29
	Our full method	**29.56**
Chai et al's method [10]		29.13
TempoGAN [33]	3D positions	26.24
	Euler rotation vectors	26.20

2. http://www.cemyuksel.com/research/hairmodels

Fig. 11: **Generalization.** Our method produces reasonable weights according to different hairstyles that vary in factors like distributions (top row), growth directions (middle row), and parting styles (bottom row). Swapping the weights of hairstyle A (leftmost column) and hairstyle B (rightmost column) for interpolation leads to severe artifacts (the two columns in the middle). This observation proves the adaptability of our neural interpolator.

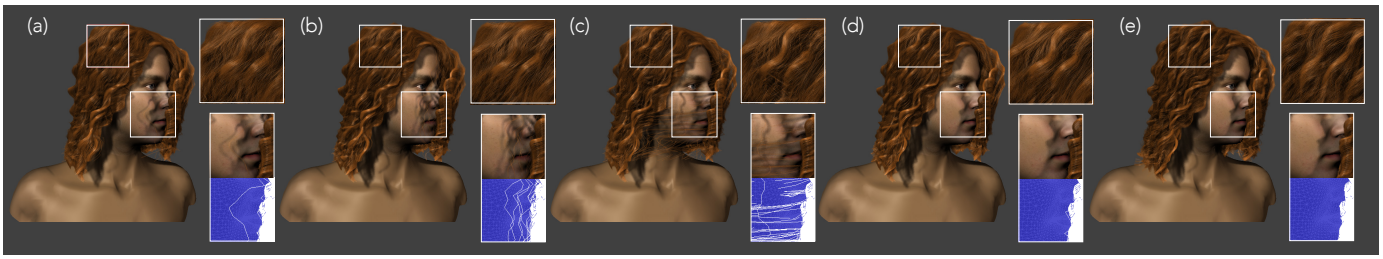


Fig. 12: **Ablation Study.** (a) $\mathcal{L}_{data}$ only. (b) Without the modal snapping term $\mathcal{L}_{snap}$. (c) Without the non-stretchable term $\mathcal{L}_{len}$. (d) Our full loss function. (e) The ground-truth.

capable of synthesizing according to the local features.

We conduct a hairstyle mirror test to verify the necessity and efficacy of the local adaptivity (Figure 11 top). Precisely, we mirror a hairstyle A about the plane of symmetry to generate another hairstyle B. Our neural interpolator predicts reasonable weights and produces realistic hairs on A and B, respectively. However, directly applying the predicted weights to the mirrored hairstyle, *i.e.*, ω^A to B and ω^B to A, lead to degraded results with severe artifacts, which justifies the necessity of the local adaptivity. Our neural interpolator naturally generalizes to free swaps and mirrors of subregions, i.e., the change of distribution. We leverage on this superiority to realize the hairstyle editing tool in Section 5.9.

Figure 11 (middle and bottom rows) shows the generalization abilities on hairstyles under the change of growth direction and parting style, respectively. Again, our neural interpolator is capable of producing realistic results on all the hairstyles, ranging from wavy to backcombed and from middle-parting to side-parting. The artifacts appear when we directly transfer the weights predicted on one hairstyle to another for interpolation, which demonstrates that the neural interpolator is capable of adapting the weights prediction process to the local features extracted from the rest shape and guide hairs. See the supplemental for more tests under curliness and length variations.

5.4 Ablation Study

Figure 12 shows the inference results of the neural interpolator trained with different loss terms discarded. A simple L1 loss leads to over-smoothing hairs that lack salient details like the clusters of strands. Meanwhile, intersections appear between the hairs and the head. The modal snapping term $\mathcal{L}_{snap}$ is vital to break the symmetry, i.e., encouraging the normal hairs to follow a specific cluster of guide hairs. Without it, intersection artifacts quickly emerge due to an unbiased interpolation. According to our observation, this loss term also strengthens the clustering effect and brings in

more hair details. The hair strands are inherently stretch-resistant in the real world. Without the loss term $\mathcal{L}_{len}$, zigzag artifacts, i.e., visible changes of segment lengths, would appear due to the conflict between the data term and the snap term. On the contrary, our complete loss function (Equation 4) generates satisfactory details without introducing any noticeable artifacts.

The temporal constraint is the key to preventing a sudden change of the velocity field during simulation, which leads to flickering artifacts of hair strands (see the video). Figure 13 shows that many discontinuous points would emerge if we exclude $\mathcal{L}_{tempo}$ from the loss function.

The data augmentation strategy remarkably increases the generalization capability because of the promoted local adaptivity, obtained from the enriched information of mutual relationships. Figure 14 shows the appearance of intersection artifacts when we train the network without using the data augmentation. The complex mutual interactions between adjacent hair strands make a single set of guide hair samplings inadequate to capture all the information.

5.5 Comparison to Skinning with Static Weights

Traditional skinning methods precompute a fixed set of weights and apply them to blend the transformations at handles. Therefore, the weights are *static* to the data. In contrast, our neural interpolator has a weight predictor component that generates weights according to the input data, i.e., they are *dynamic* to the guide hairs at each frame.

Here we compare our neural interpolator with two different hair skinning methods using static weights. The first method formulates the computation of the static weights as a quadratic programming problem, which directly minimizes the L2 loss between $\mathbf{s}$ and $\hat{\mathbf{s}}$ subject to the non-negativity and partition-of-unity constraints of ω. Rather than training a network to predict the weights, here, the weights ω themselves are the globally optimized variables (independent of per-frame guide hairs). The second one is the state-of-the-art hair skinning method [10], which builds a graph structure of all hair particles and designs partitioning and clustering algorithms to jointly compute both the guide hairs and the interpolation weights that best fit the simulation data. In experiments, the two methods using static weights tend to generate over-smoothing results that lack hair details like the strand aggregations and salient curvatures (Figure 15). Moreover, the 3D positions of their output have a lower distance to the ground-truth compared with our neural interpolator, which reflects the fact that the MAE alone is not a clear indicator for human perception of hair similarity.

To further demonstrate the necessity of dynamic skinning weights, we show the hair states at two moments of the ground-truth simulation in Figure 16 (left). At the moment t_1, the normal hair strand goes along with the guide hair g_1; but at the moment t_2, it is bumped into the cluster of another guide hair g_2 due to a violent collision between the hair strands and the head. A static set of weights cannot express this changing effect during the simulation process. In contrast, Figure 16 (right) shows the dynamic weights predicted by our neural interpolator for this sequence. The weights intuitively adapt with the guide hair states to follow the ground-truth data.

The above comparison is executed on a single hairstyle. To fully demonstrate the merit of our CNN-based dynamic skinning, we also compare with the state-of-the-art method

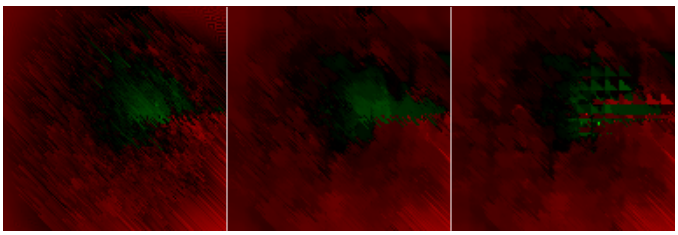


Fig. 13: **Velocity Field in a uv Plane.** (left) The ground truth. (middle) Using the temporal loss term. (right) Without the temporal loss term. Note the clumped discontinuities.

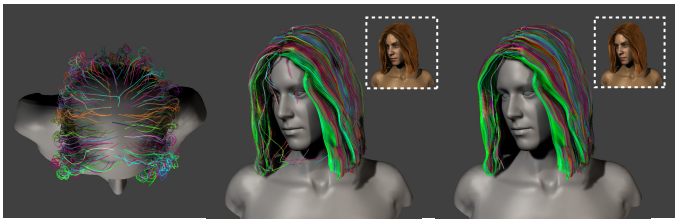


Fig. 14: **Data Augmentation Comparison.** (left) Complex mutual interactions and arrangements between hair strands. (middle) Without data augmentation. (right) With data augmentation.

Fig. 15: **Comparison to Skinning Methods with Static Weights.** Compared with the straightforward quadratic programming method (leftmost) and the cutting-edge data-driven method [10] (middle left), our dynamic weights prediction and interpolation produce much more realistic hair details (middle right), and the result is visually much similar to the ground-truth (rightmost).

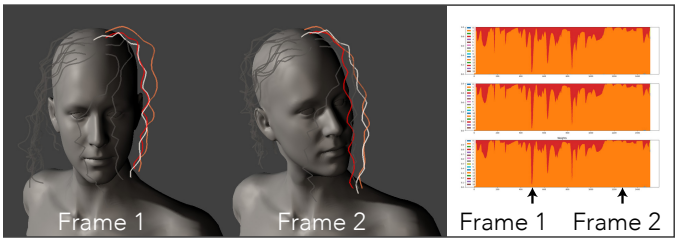


Fig. 16: **Dynamic Weights of a Hair Strand.** (left) The hair states at two moments. During the process, the white normal hair strand leaves the red guide strand and embraces the orange one. The gray guide hair strands contribute very little to the interpolation. (right) Dynamic weights of three particles in the white normal hair strand.

[10] on multiple hairstyles, as shown in Figure 17. Specifically, we use a training set of two hairstyles and test on another unseen hairstyle. The model presented by Chai *et al.* [10] fails to recover all the training data and produces artifacts on the test data. The strategy of static weights restricts the representation ability of the model. Thus the fitting quality deteriorates quickly when more distinct hairstyles are involved in the training process. In contrast, our model fits all the training hairstyles more accurately and generalizes well to the test hairstyle.

5.6 Comparison to Other Architectures

Previous works have presented end-to-end methods working well for physics-based simulation, e.g., tempoGAN [33] for the smoke. Considering this fact, we execute two experiments that apply the tempoGAN architecture to regress the hair positions and deformations respectively and compare them with our neural interpolator. We represent the hair position by the Euclidean coordinates of the particles and represent the deformation by the concise rotation vectors (multiplying the angle with a unit vector of the axis) of each hair segment. Figure 18 shows the output results of these two end-to-end methods as well as the neural interpolator. Compared with them, our approach not only exhibits a higher capability in capturing realistic details but also saves lots of computations. There is a critical difference between our problem and the smoke simulation. The hair data is strongly anisotropic due to its strand structure, but the smoke data is inherently isotropic. This difference partially explains why tempoGAN architecture cannot work well on hair data. In contrast, our predict-and-interpolate strategy obeys the structural anisotropy of the hair data by design. Moreover, the interpolation strategy effectively regularizes the output space, which makes our neural network much easier to converge and much more stable to generalize.

Fig. 17: **Comparison to the State-Of-The-Art on Multiple Hairstyles.** From left to right: Chai *et al.* [10], our method, and the ground-truth. Results of [10] have artifacts on both training data (top row) and test data (bottom row).

Fig. 18: **Comparison with Alternative Architectures.** Compared with the tempoGAN-like networks directly producing the 3D positions (leftmost) and the Euler rotation vectors (middle left), our neural interpolator robustly produces perceptually realistic hairs (middle right) that are much more faithful to the ground truth (rightmost).

5.7 Locality Assumption and Non-Local Interactions

Our method relies on the locality in the uv plane for strand interpolation and hairstyle generalization. In the real world, the interactions between hair strands can be non-local, especially for long hairstyles. Though our CNN-based neural interpolator merely uses a moderate receptive field, hair strands far away in the uv plane (outside of each other's receptive field) can still interact in the final results. The reason is mainly due to the presence of the guide hairs. Once an interaction happens between remote hair strands in the full simulation data, it is very likely that their nearby guide hairs also interact (see Figure 19b). In such cases, the physics-based simulation of the guide hairs at runtime preserves those remote interactions and further transfers them to the normal hair interpolations. According to our statistics, the non-local interactions occupy less than 30% of all the hair interactions. Our method succeeds in preserving about half of the non-local interactions in the full simulation data (see Figure 19a). We believe that despite its simplifying assumption, our system produces practically-promising results that preserve the main hair interactions without losing much dynamic detail. Furthermore, our method achieves this level of detail within a limited time budget, which is essential for most real-time and interactive applications.

5.8 Performance

By leveraging the carefully designed network architecture and the GPU computation, our CNN-integrated simulation framework offers an interactive speed on hairstyles of up

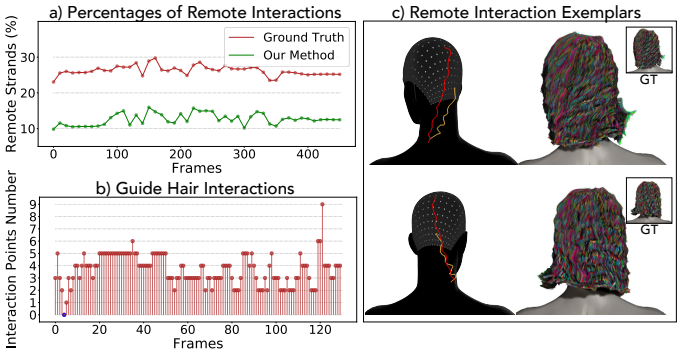


Fig. 19: **Non-local hair Interactions.** a) Percentages of remote hair interactions in ground-truth and our results. For each hair particle, we count its closest 50 particles in the Euclidean configuration space for interactions. Two strands outside of each other's 32×32 local region in the uv space are regarded as remote. b) The interaction amount between the guide hairs in the neighborhood of two remote interacting normal strands. c) Exemplars of remote interactions between two normal strands during the inference phase.

TABLE 2: Per-frame Runtime Statistics

Method		Time in seconds (FPS)	
		128×128	256×256
Our	Guide Hair Simulation	0.0084	
	Neural Interpolator	0.024	0.12
	Fine-scale Displacement	0.0028	0.0056
	Data Transmission	0.0068	0.026
	Total	0.042 (23.7)	0.16 (6.4)
	Full Hair Simulation	0.58 (1.7)	2.5 (0.4)

to 256×256 resolution, i.e., more than 60K strands. Table 2 shows the statistics of decomposed and total runtime. Among all the online computations, the neural interpolator is the main bottleneck ($\approx 65\%$ of the total time), and the data transmission between GPU and main memory takes more time than the fine-scale displacement generator. Compared with the full simulation, our method produces visually similar hair details while obtaining about 14 and 16 times of speedup on resolution 128×128 and 256×256, respectively.

5.9 Application

To demonstrate the usage, we build a performance-driven digital avatar system by integrating our interactive hair simulation method with a video-based facial tracking module [46]. Figure 20 shows a real human actor driving the digital avatar, whose hair simulation is controlled by the rigid head transformation tracked in real-time. Please see the video for a live demo.

Besides that, we also build a hairstyle editing tool to illustrate the potential of our simulation method in interactive applications. With this tool, an average user can interactively change the length, curliness, growth direction, and parting style of the hairs, and even composite a novel one by copy-and-paste from multiple hairstyles. Our simulator provides the user with immediate feedback on the dynamic behavior after the hair editing. Please see Figure 21 and the video for more details.

6 Conclusion, Limitation and Future Work

We have introduced a CNN integrated framework for simulating multiple hairstyles. It yields high-fidelity hair details

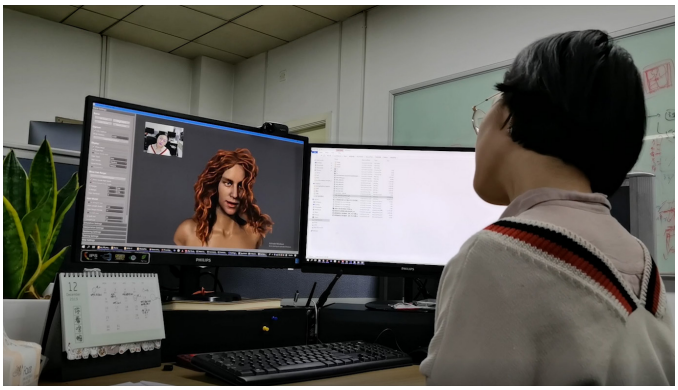


Fig. 20: **Performance-driven digital avatar.** We drive the hair simulation by the average user's head motions tracked in real-time.

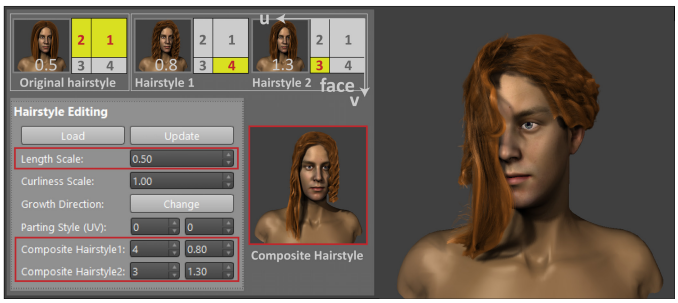


Fig. 21: **Online hairstyle editing.** The user can interactively adjust the hairstyle in a control panel and see the prompt feedback of its new dynamic behavior.

in an interactive speed. By pre-training on representative hairstyles, our neural interpolator and fine-scale displacement generator robustly generalize to novel hairstyles and poses. We thus hope our approach further advances the deployment of hair simulation into interactive AR and VR applications.

Our method also has several limitations. We uniformly sample a fixed number of particles along each hair strand and execute the interpolation in every uv-plane of the grid. This interpolation strategy relies on the assumption that nearby hair strands have similar lengths. Otherwise, the adjacent particles in the same uv-plane could be far apart in the 3D space, which makes their interpolation meaningless. In most practice, this assumption is reasonable, but a better particle sampling/interpolation strategy should enter when it is not the case.

Our loss function requires an explicit regularization for length preservation. It is possible to entirely remove this term by choosing another interpolation space. Currently, we interpolate the vertex coordinates for its simplicity and generality, but the whole method is not restricted to this representation. For example, the curvature-twist representation for super-helices [19] builds the inextensibility constraint into the inherent formulation. Interpolation in this curvature space is an exciting direction to explore, while the neural interpolator may need dedicated controls to suppress the potential artifacts brought by such interpolations.

It could be beneficial to make the dynamics update two-way, i.e., transferring information from the high-res output of the network back to the low-res guide hairs simulation, at each timestep to increase its fidelity. For example, physical interactions like the collision forces could be estimated and transferred, at the cost of extra online computations.

Our discriminator for fine-scale displacement generator uses manually designed features as the input and classify the real/fake data in a 3D sense. Ideally, it makes more sense to judge from the 2D image directly whether a hair model is perceptively similar to the ground-truth or not. An efficient differentiable renderer dedicated to hairs is thus required for back-propagating the gradient information.

So far, we could not achieve the interactive performance on extremely high-resolution hairstyles, e.g., 512×512. We could obtain a more lightweight network by using the smart weight and unit pruning methods to accelerate the inference speed.

We generate the full simulation data by using the same

physical parameters for the same hairstyles. The variance of data would significantly increase when we use different parameters. Training a representation network competent for this data, e.g., by conditioning on the additional input of physical parameters, deserves further exploration.

Last but not least, how to extend our method to take into account more complex mechanics such as the advanced frictional contacts [47] and liquid-hair interactions [5] should be an exciting research direction.

APPENDIX A
SCALP PARAMETERIZATION

According to the work [48], the scalp surface is considered as the upper half of a sphere approximately. Then the scalp surface parameterization is defined as

$$u = \arccos \frac{\hat{x}}{\sqrt{\hat{x}^2 + (\hat{y}+1)^2}}$$
$$v = \arccos \frac{\hat{z}}{\sqrt{\hat{z}^2 + (\hat{y}+1)^2}},$$

where $(\hat{x}, \hat{y}, \hat{z}) = \frac{\mathbf{p}-\mathbf{o}}{\|\mathbf{p}-\mathbf{o}\|}$ is the projection of a scalp point $\mathbf{p}$ onto the unit sphere centered at $\mathbf{o}$, whose south pole point is at $(0, -1, 0)$ in the local system. Note that we discard the parameter w defined in [48].

APPENDIX B
DETAILS OF DATA

Here we summarize the details of the hair data used in this paper. We use ten different hairstyles for training. Besides these hairstyles, we add another nine different hairstyles for testing the learned model. The head motions for training and testing are mutually exclusive.

		Train	**Test**
#Hairstyles	Curly	5	5+1
	Straight	5	5+1
	Wavy	-	1
	wCurly	-	1
	Straight to Curly	-	1
	Half Curliness	-	1
	Random Growth Directions	-	2
	Side Part	-	1
#Frames of Head Motion	Random	12000	1500
	Manually Specified	-	460
#DataFrames in Total		120000	37240

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments. This research is partially supported by National Natural Science Foundation of China (61772024, 61732016, 61890954).

REFERENCES

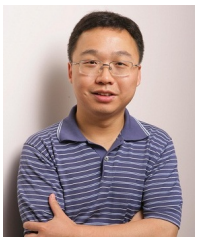
[1] S. Saito, L. Hu, C. Ma, H. Ibayashi, L. Luo, and H. Li, "3d hair synthesis using volumetric variational autoencoders," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, p. 208, 2019.
[2] M. Zhang and Y. Zheng, "Hair-gan: Recovering 3d hair structure from a single image using generative adversarial networks," *Visual Informatics*, 2019.
[3] G. Daviet, F. Bertails-Descoubes, and L. Boissieux, "A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics," in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6. ACM, 2011, p. 139.
[4] D. M. Kaufman, R. Tamstorf, B. Smith, J.-M. Aubry, and E. Grinspun, "Adaptive nonlinearity for collisions in complex rod assemblies," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 123, 2014.
[5] Y. R. Fei, H. T. Maia, C. Batty, C. Zheng, and E. Grinspun, "A multi-scale model for simulating liquid-hair interactions," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 56, 2017.
[6] E. Plante, M.-P. Cani, and P. Poulin, "A layered wisp model for simulating interactions inside long hair," in *Computer Animation and Simulation 2001*. Springer, 2001, pp. 139–148.
[7] J. T. Chang, J. Jin, and Y. Yu, "A practical model for hair mutual interactions," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2002, pp. 73–80.
[8] S. Hadap and N. Magnenat-Thalmann, "Modeling dynamic hair as a continuum," in *Computer Graphics Forum*, vol. 20, no. 3. Wiley Online Library, 2001, pp. 329–338.
[9] Y. Bando, B.-Y. Chen, and T. Nishita, "Animating hair with loosely connected particles," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 411–418.
[10] M. Chai, C. Zheng, and K. Zhou, "A reduced model for interactive hairs," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 124, 2014.
[11] ——, "Adaptive skinning for interactive hair-solid simulation," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 7, pp. 1725–1738, 2016.
[12] K. Ward, F. Bertails, T.-Y. Kim, S. R. Marschner, M.-P. Cani, and M. C. Lin, "A survey on hair modeling: Styling, simulation, and rendering," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 2, pp. 213–234, 2007.
[13] F. Bertails, S. Hadap, M.-P. Cani, M. Lin, T.-Y. Kim, S. Marschner, K. Ward, and Z. Kačić-Alesić, "Realistic hair simulation: animation and rendering," in *ACM SIGGRAPH 2008 classes*. ACM, 2008, p. 89.
[14] R. E. Rosenblum, W. E. Carlson, and E. Tripp III, "Simulating the structure and dynamics of human hair: modelling, rendering and animation," *The Journal of Visualization and Computer Animation*, vol. 2, no. 4, pp. 141–148, 1991.
[15] B. Choe, M. G. Choi, and H.-S. Ko, "Simulating complex hair with robust collision handling," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2005, pp. 153–160.
[16] A. Selle, M. Lentine, and R. Fedkiw, "A mass spring model for hair simulation," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, p. 64, 2008.
[17] K.-i. Anjyo, Y. Usami, and T. Kurihara, "A simple method for extracting the natural beauty of hair," *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, pp. 111–120, 1992.
[18] A. Daldegan, N. M. Thalmann, T. Kurihara, and D. Thalmann, "An integrated system for modeling, animating and rendering hair," in *Computer Graphics Forum*, vol. 12, no. 3. Wiley Online Library, 1993, pp. 211–221.
[19] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, and J.-L. Lévêque, "Super-helices for predicting the dynamics of natural hair," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 1180–1187.
[20] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," in *ACM transactions on graphics (TOG)*, vol. 27, no. 3. ACM, 2008, p. 63.
[21] A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran, "Detail preserving continuum simulation of straight hair," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 62.
[22] A. Derouet-Jourdan, F. Bertails-Descoubes, G. Daviet, and J. Thollot, "Inverse dynamic hair modeling with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 159, 2013.

[23] L. Petrovic, M. Henne, and J. Anderson, "Volumetric methods for simulation and rendering of hair," *Pixar Animation Studios*, vol. 2, no. 4, 2005.
[24] M. Müller, T.-Y. Kim, and N. Chentanez, "Fast simulation of inextensible hair and fur." *VRIPHYS*, vol. 12, pp. 39–44, 2012.
[25] C. Yuksel, S. Schaefer, and J. Keyser, "Hair meshes," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5. ACM, 2009, p. 166.
[26] K. Wu and C. Yuksel, "Real-time hair mesh simulation," in *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2016, pp. 59–64.
[27] K. Ward, M. C. Lin, L. Joohi, S. Fisher, and D. Macri, "Modeling hair using level-of-detail representations," in *Proceedings 11th IEEE International Workshop on Program Comprehension*. IEEE, 2003, pp. 41–47.
[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
[29] S. W. Bailey, D. Otte, P. Dilorenzo, and J. F. O'Brien, "Fast and deep deformation approximations," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 119, 2018.
[30] L. Liu, Y. Zheng, D. Tang, Y. Yuan, C. Fan, and K. Zhou, "Neuroskinning: automatic skin binding for production characters with deep graph networks," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 114, 2019.
[31] R. Luo, T. Shao, H. Wang, W. Xu, X. Chen, K. Zhou, and Y. Yang, "Nnwarp: Neural network-based nonlinear deformation," *IEEE transactions on visualization and computer graphics*, 2018.
[32] L. Fulton, V. Modi, D. Duvenaud, D. I. Levin, and A. Jacobson, "Latent-space dynamics for reduced deformable simulation," vol. 38, no. 2, pp. 379–391, 2019.
[33] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 95, 2018.
[34] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.
[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
[36] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[37] M. Chai, T. Shao, H. Wu, Y. Weng, and K. Zhou, "Autohair: Fully automatic hair modeling from a single image," *ACM Transactions on Graphics*, vol. 35, no. 4, 2016.
[38] Y. Zhou, L. Hu, J. Xing, W. Chen, H.-W. Kung, X. Tong, and H. Li, "Hairnet: Single-view hair reconstruction using convolutional neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 235–251.
[39] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: learning dynamic renderable volumes from images," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 65, 2019.
[40] K. Olszewski, D. Ceylan, J. Xing, J. Echevarria, Z. Chen, W. Chen, and H. Li, "Intuitive, interactive beard and hair synthesis with generative models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7446–7456.
[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, 2017, pp. 214–223.
[43] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[44] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[46] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3d shape regression for real-time facial animation," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 41, 2013.
[47] C. Jiang, T. Gast, and J. Teran, "Anisotropic elastoplasticity for cloth, knit and hair frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 152, 2017.
[48] L. Wang, Y. Yu, K. Zhou, and B. Guo, "Example-based hair geometry synthesis," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, p. 56, 2009.

Qing Lyu received the bachelor's degree in Digital Media Technology from Zhejiang University in 2016. Currently, she is working toward the PhD degree at the State Key Lab of CAD&CG, Zhejiang University. Her research interests include visual computing and computer graphics.

Menglei Chai is a Senior Research Scientist at Snap Inc. He received his Ph.D. and B.Eng. in Computer Science from Zhejiang University in 2017 & 2011. His research interest is Computer Vision and Graphics, especially in photo manipulation and physics-based simulation.

Xiang Chen is an Associate Professor in the State Key Lab of CAD&CG, Zhejiang University. He received his Ph.D. in Computer Science from Zhejiang University in 2012. His current research interests mainly include fabrication-aware design, physics-based simulation, image analysis/editing, shape modeling/retrieval and computer-aided design.



Kun Zhou is a Cheung Kong Professor in the Computer Science Department of Zhejiang University and the Director of the State Key Lab of CAD&CG. He received my BS degree and PhD degree in computer science, both from Zhejiang University. After graduation he spent six years with Microsoft Research Asia, and was a lead researcher of the graphics group before moving back to Zhejiang University. He was named one of the world's top 35 young innovators by MIT Technology Review in 2011, and was elected as an IEEE Fellow in 2015.

Real-Time Hair Simulation with Neural Interpolation

SUPPLEMENTAL MATERIAL

Qing Lyu* Menglei Chai† Xiang Chen* Kun Zhou*

* Zhejiang University † Snap Research

1 Efficacy of the Displacement Generator on Different Resolutions

The fine-scale displacement generator produces hair details and superimposes them on the neural interpolator's output. We keep using 16×16 guide hairs and test the displacement generator's efficacy on different normal hair resolutions. When the resolution is relatively low, e.g., 64×64, the neural interpolator's output is sufficiently close to the ground truth, and the fine-scale displacement generator only makes a limited promotion of the hair details (see Figure 1). When the resolution of normal hairs goes up to 256×256, the neural interpolator's output is rather smooth, and the fine-scale displacement generator considerably promotes the realistic details (see Figure 2).

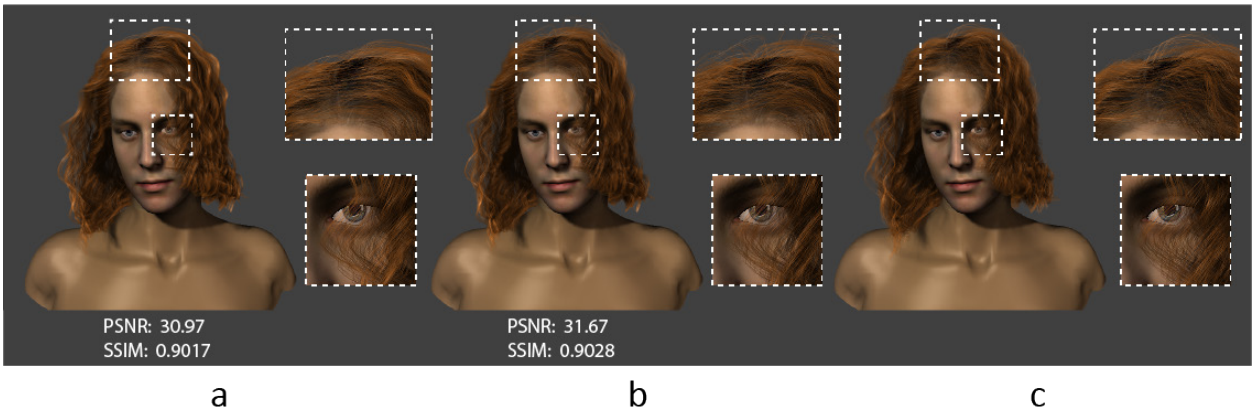


Figure 1: **Displacement generation on** 64×64 **normal hairs.** Compared with the ground-truth (right), the hair details produced by the fine-scale generator (middle) show a limited promotion of the neural interpolator's output (left).

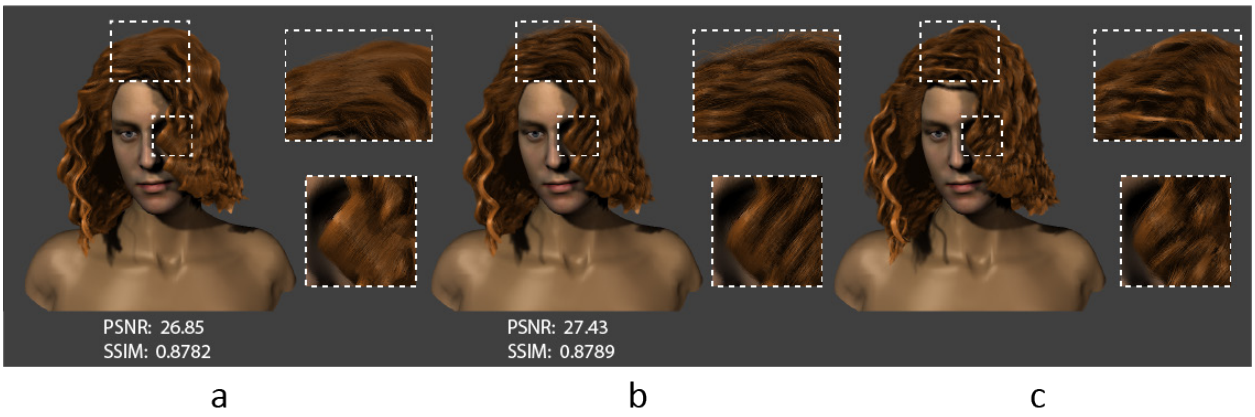


Figure 2: **Displacement generation on** 256×256 **normal hairs.** Compared with the ground-truth (right), the hair details produced by using the fine-scale generator (middle) is visually more realistic than that of without using it (left).

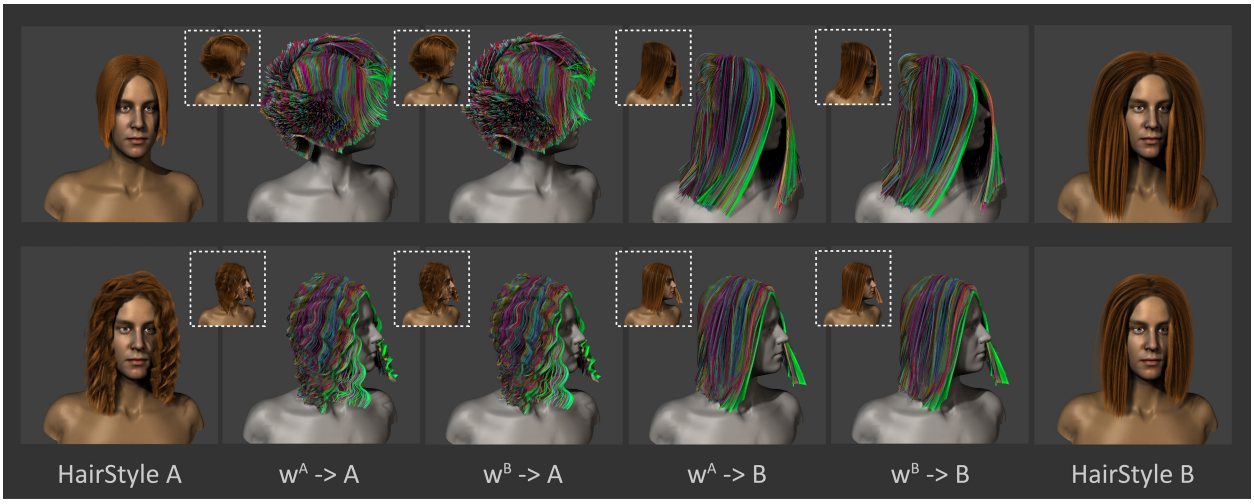


Figure 3: **Generalization test.** Our method produces reasonable weights according to different hairstyles that vary in factors like hair length (top row) and curliness (bottom row).

Figure 4: **Robustness test.** Compared with the ground-truth (top row), our neural interpolator produces stable and plausible results (bottom row) under extreme head motions.

2 Generalization Experiments on Hair Length and Curliness

Besides the generalization tests in the paper, we also test generalization abilities on hairstyles (see Figure 3) under the change of the hair length (top row) and curliness (bottom row), respectively. Our neural interpolator is capable of producing realistic results on all the hairstyles, ranging from short to long and from curly to straight. Since the growth directions of these hairstyles are almost consistent, there is little artifact but only degradation of hair details when swapping the predicted weights of hairstyle A and B for interpolation.

3 Extreme Motion

To explore the robustness of the neural interpolator, we prepare a sequence of extreme head motions (5,000 frames) and experiment on it. The hair interpolations are consistently stable and plausible (see Figure 4 for some snapshots).

4 Strand Level vs. Particle Level Interpolation

In our neural interpolator, we choose to predict distinct weights for individual particles and execute the interpolation on the particle level. Another option is interpolating the hair states by using shared weights for particles of the same strand (strand-level), providing a stronger regularization. Theoretically, particle-level interpolation unbinds the particles of the same strand and provides flexibilities for individual particles. In experiments, it shows a better capability of details preservation (see Figure 5) at the risk of longer training time.

Figure 5: **Strand-level vs. Particle-level interpolation.** Compared with the ground-truth (bottom row), particle-level interpolation (middle row) preserves hair details more faithfully than strand-level interpolation (top row).

Figure 6: **Comparison to the State-Of-The-Art on Another Single Hairstyle.** From left to right: [chai et al. 2014], our method, and the ground-truth.

5 Comparison to the State-Of-The-Art Method

Figure 6 shows the comparison to the state-of-the-art method [chai et al. 2014] on another single hairstyle. Again, salient hair features like the clustering effects are more faithfully preserved in our results.

6 Details of Network Architecture

We list the architecture details of W, F, and D in the following table. The notations to specify our networks include:

- $\mathsf{C}(\mathsf{K}, \mathsf{S}, \mathsf{a})$ denotes a convolutional layer with kernel size $(\mathsf{K}, \mathsf{K}, \mathsf{K})$, stride $(\mathsf{S}, \mathsf{S}, 1)$, and activation a.
- $\mathsf{T\_C}(\mathsf{K}, \mathsf{S}, \mathsf{a})$ denotes a transposed convolution in a similar setting.
- $\mathsf{x1}[\mathsf{C}, \mathsf{IN}] + \mathsf{x2}[\mathsf{C}, \mathsf{IN}], \mathsf{LReLU}$ denotes a skip connection $[(\mathsf{x1} \rightarrow \mathsf{C} \rightarrow \mathsf{IN}) + (\mathsf{x2} \rightarrow \mathsf{C} \rightarrow \mathsf{IN})] \rightarrow \mathsf{LReLU}$, where $\rightarrow$ means the data flow.
- IN and LReLU represent the instance normalization and the leaky version of a rectified linear unit, respectively.

Name	**Structure**
W_θ (Section 4.2.1 in the paper)	
$\mathbf{g}$(input)	size : $(16, 16, 25, 3)$
$\bar{\mathbf{s}}$(input)	size : $(128, 128, 25, 3)$
n_r1	$\bar{\mathbf{s}}$[C$(5, 2, 8)$, IN, LReLU]
n_r2	n_r1[C$(5, 2, 16)$, IN, LReLU]
$\bar{\mathbf{s}}^*$(output)	n_r2[C$(5, 2, 32)$, IN, LReLU] size : $(16, 16, 25, 64)$
n_g1	$\mathbf{g}$[C$(5, 1, 32)$, IN] + $\mathbf{g}$[C$(1, 1, 32)$, IN], LReLU
n1	Concatenate(n_g1, $\bar{\mathbf{s}}^*$)
n2	n1[T_C$(4, 2, 64)$, IN] + n_r2[C$(1, 1, 64)$, IN], LReLU
n3	n2[T_C$(4, 2, 32)$, IN] + n_r1[C$(1, 1, 32)$, IN], LReLU
ω(output)	n3[T_C$(4, 2, 16)$, IN] + $\bar{\mathbf{s}}$[C$(1, 1, 16)$, IN], Softmax size : $(128, 128, 25, 16)$
F_β (Section 4.3.1 in the paper)	
n_rf	$\bar{\mathbf{s}}^*$[C$(1, 1, 1)$, IN]
in1	Concatenate($\mathbf{g}$, n_rf)
n_res1	in1[C$(5, 1, 32)$, IN] + $\mathbf{g}$[C$(1, 1, 32)$, IN], LReLU
n_res2	n_res1[T_C$(5, 4, 16)$, IN]
(output)	n_res2[T_C$(5, 2, 3)$, IN] size : $(128, 128, 25, 3)$
D_γ (Section 4.3.2 in the paper)	
in_dis(input)	size : $(128, 128, 25, 5)$
n_d1	in_dis[C$(4, 2, 32)$, IN, LReLU]
n_d2	n_d1[C$(4, 2, 64)$, IN, LReLU]
n_d3	n_d2[C$(4, 2, 128)$, IN, LReLU]
n_d4	n_d3[C$(4, 1, 256)$, IN, LReLU]
(output)	n_d4[FullyConnected] size : (1)