A Reduced Model for Interactive Hairs

Menglei Chai* Changxi Zheng† Kun Zhou*‡

*State Key Lab of CAD&CG, Zhejiang University †Columbia University

(a) (b) (c) (d) (e)

Figure 1: A reduced skinning model of 150K hairs for interactive simulation: *(a) We visualize the guide hairs and their weights on other hairs using a colormap. With 150K hair strands, our reduced simulation runs at 40ms per frame. Two of the simulated frames are shown in (b) and (d). As a comparison of the simulation quality, we show the same frames generated by a full simulation (30-60 seconds per frame) in (c) and (e). Our reduced model exhibits comparable hair motions and details as captured by the full simulation.*

Abstract

Realistic hair animation is a crucial component in depicting virtual characters in interactive applications. While much progress has been made in high-quality hair simulation, the overwhelming computation cost hinders similar fidelity in realtime simulations. To bridge this gap, we propose a data-driven solution. Building upon precomputed simulation data, our approach constructs a reduced model to optimally represent hair motion characteristics with a small number of guide hairs and the corresponding interpolation relationships. At runtime, utilizing such a reduced model, we only simulate guide hairs that capture the general hair motion and interpolate all rest strands. We further propose a hair correction method that corrects the resulting hair motion with a position-based model to resolve hair collisions and thus captures motion details. Our hair simulation method enables a simulation of a full head of hairs with over 150K strands in realtime. We demonstrate the efficacy and robustness of our method with various hairstyles and driven motions (e.g., head movement and wind force), and compared against full simulation results that does not appear in the training data.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: hair simulation, data-driven animation, collisions

Links: DL PDF WEB

*cmlatsim@gmail.com, kunzhou@acm.org

†cxz@cs.columbia.edu

‡Corresponding author

1 Introduction

With a striking development of realistic hair models, hair simulation has popularized itself in computer generated movies. Although equally desired in computer games and interactive applications, realtime hair simulation has been much less explored. The main challenge is due to a high complexity of realistic hair models. For instance, a young adult typically has more than 100K hair strands. Realistic simulation of such amount of hair strands must consider hair-hair and hair-body interactions, yielding an overwhelming computation cost.

Our goal is to develop a fast hair simulation method that is able to capture realistic hair motion details with interactive performance. Although making up-front simplification is important for blazing performance, we prefer the choices that retains physical foundations and realism, since such a model also promises in controllability, extensibility and easy parameter setup. From this vantage point, it seems a logical strategy is to build a reduced model from physics-based hair simulation.

Our reduced model exploits the basic idea of simulating only a small set of guide hairs and interpolating the motion of a full set of hairs. While this idea has been explored in previous methods, they all share a major drawback of sacrificing hair motion details – interpolation acts like a low-pass filter that easily smears out detailed motions introduced by complex strand geometry and mutual interactions. The key challenge in designing a reduced hair model is thus how to retain the physical realism and motion details. To this end, we need to address three problems. First, we seek to effectively assign guide hairs to properly describe the "bones" of a full hair simulation. Different hair styles have different physical properties, leading to very diverse hair motion characteristics. Therefore any ad-hoc or random selection of guide hairs would be incapable of producing comparable motion details as captured in a full hair simulation. Second, it is important to decide for each single hair strand its guide hairs and assign the weights such that the interpolation results match full simulation results as well as possible. Finally, even with carefully chosen guide hairs and optimal interpolation weights, the nature of interpolation would unavoidably cause the loss of motion details, unless hair mutual interactions are taken into account.

To address these problems, we propose to train a reduced model

from a few sequences of precomputed full simulation data. Our intuition is that although possible head motions can vary tremendously, hair motions with respect to nearby hair strands are quite limited and damped by large amount of collisions. This observation suggests that it is sufficient to extract a reduced model from only a few typical head motions. We therefore develop two techniques. First, we propose an optimized *hair skinning model* with carefully selected guide hairs based on the training data. We construct a graph that reflects the local motion similarity of hair strands in the training data and formulate guide hair selection as a graph partitioning problem, which clusters hair strands into a few groups with coherent motions. We then select a representative strand as a guide hair from each group. The interpolation weights for each hair particle are then optimized by solving a linearly constrained least-square problem.

Hair motions interpolated using the skinning model lacks details, because it ignores collisions. Our second technique is to further correct the resulting hair motions to respect collisions using a position-based correction model. To this end, we again exploit the training data and cluster hair particles into weakly coupled groups in the preprocess. At runtime, we use Jacobi iterations to resolve collisions in each group in parallel. Because of the weak coupling discovered from training data, only a small number of iterations are needed to capture plausible motion details.

Contributions. The core contribution of our work is a fast reduced hair simulation model based on a small set of training data, together with a collision correction algorithm to capture detailed runtime hair motions. We show that this model produces very similar results as captured by a full simulation, but with significantly higher performance (see Figure 1).

Furthermore, to demonstrate practical application of our method, we build a realtime hair simulation system incorporating advanced hair rendering and facial tracking. This system is able to simulate up to 150K hair strands entirely in realtime, proving the usability of our method in computer games and interactive applications, in which the total computing power has to be shared with other components. To the best of our knowledge, this level of realtime performance under a full integration of realistic hair simulation, rendering and facial animation has not yet been achieved in previous works.

2 Related Works

Hair Simulation. Realistic hair modeling and animation has long been an important research field [Ward et al. 2007], starting from the early work [Rosenblum et al. 1991; Anjyo et al. 1992] for animating individual hair strands without taking into account hair mutual interactions. For high-quality hair animation, simulating every single strand is advantageous for capture hair motion details, as advocated by [Selle et al. 2008]. Dynamic models for single hair strand have been extensively studied to plausibly capture real-world strand behaviors [Bertails et al. 2006; Selle et al. 2008; Bergou et al. 2008; Casati et al. 2013]. Based on these models, methods that operate on the strand level and take into account complex hair interactions can yield high-fidelity results. Selle et al. [2008] modeled hair interaction as a combination of strand stiction and geometric collision, successfully resolving dynamical change of wisp structures that are lacking in clump approaches. Daviet et al. [2011] developed a hybrid approach that robustly simulate the Coulomb friction effects between hair fibers. Unfortunately, all these methods focus on capturing motion details and thus requires a large computation budget to run offline. The high computation cost renders them impractical for interactive applications.

Focusing more on the high-performance simulation of hairs, various simplified representations have been proposed to accelerate hair mutual interactions. One common idea is to implicitly resolve hair interactions with continuum dynamics by exploiting the properties of volume-preserving and local motion coherence. For example, Hadap and Magnenat-Thalmann [2001] applied fluid dynamic models to resolve hair collisions; and Bando et al. [2003] modeled hairs as loosely connected particles that are animated in a continuum way. Aiming at stylized hair, some methods [Petrovic et al. 2005] used volumetric structures to filter hair velocity and density inside. A similar approach was proposed by [Müller et al. 2012] to mimic hair-hair repulsion efficiently. To enhance the hair details McAdams et al. [2009] proposed a hybrid fluid solver in the hair simulation pipeline.

Perhaps the most relevant hair representation to our approach is the aggregate clump models, in which strands are interpolated from a few simulated guide strands, and hair interactions are handled on a volumetric grid [Tariq and Bavoil 2008], or processed by simplified primitives such as strips [Chang et al. 2002], cylinders [Choe et al. 2005; Plante et al. 2001], and spheres [Iben et al. 2013]. Adaptive clump models [Bertails et al. 2003; Ward and Lin 2003; Ward et al. 2003] have further extended this idea to capture more details in certain cases without sacrificing too much computational cost.

Many of these fast hair simulation methods rely on heuristic models to trade off between accuracy and performance. They exchange the simulation quality for performance. In contrast, we optimize our reduced model based upon high-resolution full simulation data to ensure detailed hair motions to be preserved better.

Data-driven Animation. Data-driven approaches are efficient alternatives to physically based simulation, especially when the target phenomena are too expensive to simulate or too complex to model properly. They have proven useful in simulating various phenomena such as deformable objects [James and Fatahalian 2003], clothes [Kavan et al. 2011] and fluids [Wicke et al. 2009].

One common data-driven strategy is to enhance coarsely simulated results using motion details extracted from high-resolution data. This approach has been successfully applied for generating detailed cloth animations [Feng et al. 2010; Wang et al. 2011; Kavan et al. 2011]. Though physical principles may not be fully satisfied, the result improvement appears plausible in many cases and the computational overhead is usually low. Meanwhile, since the low-frequency motions are not changed for detail enhancement, the resulting motions always agree with the simulated motions in large scales.

Another strategy of data-driven approach is to construct efficient reduced models for complex dynamical systems. Such data-driven reduced models have been applied for simulating deformable objects [James and Fatahalian 2003] and fluids [Treuille et al. 2006; Wicke et al. 2009]. Our hair simulation method follows this general approach. However, for hair simulation, the motion details heavily depend on hair mutual interactions, which are challenging to resolve but critical for motion details. Our approach corrects hair motions after a reduced simulation step to enhance details. Our hair skinning model is also similar in spirit to the cubature method of nonlinear deformable simulation [An et al. 2008], which exploits the spatial coherence for fast spatial integrations of internal forces. However, we are not solving any integration problem, and hair motion might not exhibit significant spatial coherence due to numerous collisions.

For reduced hair simulation, the recent work by Guan et al. [2012] proposed a multi-linear reduce model incorporating head geometry and various user-specified parameters. The proposed simulation runs in a reduced space with up to 4000 hair strands being animated in about 20ms per frame. However, one major limitation is that the runtime character motion needs to have sufficient similarity as the training data. Moreover, this method considers no hair-hair interactions, therefore lacking many details. Our method distinguishes from this approach in two aspects: first, we optimize the selection

Figure 2: Skinning interpolation: *Our reduced model interpolates a full set of hairs (right) using a few guide hairs (left) visualized with different colors. (middle) We visualize the hair blending weights of the skinning model using a color map.*

of guide hairs and their weights for affecting other hairs. As a result, our reduced model represents the local hair relationships with large-scale motions constrained by physically simulated guide hairs. This approach is flexible to incorporate various head motions and environmental forces. Meanwhile, we explicitly resolve hair-hair and hair-body interactions to recover realistic motion details.

3 Rationale and Overview

Interactive hair simulation is challenging, mainly because numerous hair strands need to be considered and intricate collisions prevail throughout the simulation. At every step, a simulator needs to detect collisions, compute contacts and internal forces, and advance every single hair strand. In other words, collision resolution are *coupled* in the hair dynamics. As a result, previous interactive hair models have to sacrifice detailed collision resolution in exchange for fast hair dynamics, and thus fail to retain hair details such as clump separation and stray hairs.

Key Idea. We propose to *decouple* the detailed collision resolution from hair dynamics. Our method first simulates hair dynamics and advance hair states using a reduced model, followed by a hair correction step that resolves collisions for hair details. This decoupled scheme enables us to build separated models for both steps. Unlike previous hair simulation methods, we optimize both our reduced hair model and hair correction scheme based on precomputed full simulation data. This idea is motivated by the observation that widespread hair collisions effectively restrict hair strands to move in a limited way with respect to their nearby hairs. Thus, a small set of full simulation data is sufficient to extract local statistical information that reflects spatial coherence of hair motions and helps to accelerate collision resolution. Concretely, our method consists of the following two components.

Offline Training. Provided a time sequence of hair motion from a full simulation, our system selects a set of representative strands based on their local motion similarity exhibited in the training data (§5). These hair strands will serve as guide hairs in our reduced hair model at runtime. Next, we learn a hair skinning model that interpolates the motions of a full set of hairs using the guide hairs (§5.2). This skinning model resembles a linear blend skinning (LBS) model [James and Twigg 2005; Kavan et al. 2011], in which the guide hairs serve as bones while the others act like skin. In addition to building the skinning model, we extract hair-hair interaction statistics from the training data and cluster hairs into weakly coupled regions (§5.4). These regions, called *hair correction groups*, are critical to accelerate our hair correction step at runtime.

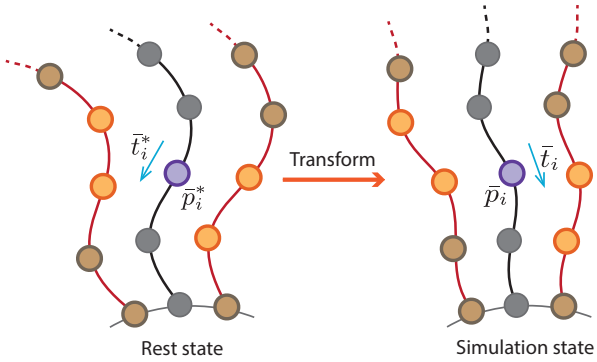


Figure 3: Hair Skinning Model: *In our hair skinning model, each normal hair particle (in purple) is interpolated from its guide hair particles (in orange). (Left) We describe the rest state of a hair particle using a tangential vector $\bar{\mathbf{t}}_i$ and particle position $\bar{\mathbf{p}}_i$ in the head's local frame of reference. (Right) We use the skinning model Eq. (2) to interpolate normal hair particles also in the head's frame of reference.*

Interactive Hair Simulation. At runtime, we directly simulate the selected guide hairs to capture general hair motions and rough collisions (§6.1). We then interpolate the motions of entire hair strands using the learned hair skinning model. These steps serve as a prediction, resulting full-resolution hair dynamics but lacking motion details. Lastly, as a correction step, we resolve detailed hair collisions. To ensure interactive performance, we propose a position-based correction model (§6.2), which resolves collisions in parallel on all hair correction groups, and converges within a very small number of iterations.

4 Reduced Hair Model

Hair Representation. In our reduced hair model, a hair strand is a B-spline curve described by a chain of particles. In addition to position $\boldsymbol{p} \in \boldsymbol{R}^3$, a hair particle also carries a tangential direction $\boldsymbol{t}$ to aid smooth interpolation in our hair skinning model (detailed later). In general, we use $\boldsymbol{s} = (\boldsymbol{p}, \boldsymbol{t})$ to denote the state of a hair particle. Since hairs are always attached to a rigid head, we put a bar over a letter (i.e., $\bar{\boldsymbol{s}} = (\bar{\boldsymbol{p}}, \bar{\boldsymbol{t}})$) to denote hair particle state in the head's *local* frame of reference. We refer $\bar{\boldsymbol{s}}$ as the local coordinate of a hair particle, and $\boldsymbol{s}$ as its world coordinate. Given a rigid head transformation $\boldsymbol{T}$, the local-to-world transformation of a hair particle is $\boldsymbol{s} = \boldsymbol{T}\bar{\boldsymbol{s}}$, in which $\boldsymbol{p}$ is translated and $\boldsymbol{t}$ is rotated by $\boldsymbol{T}$.

Similar to previous approaches [Chang et al. 2002], our reduced hair model consists of a small set of guide hairs and the rest of the hairs, which we call *normal* hairs to distinguish them from the guide hairs. To ensure interactive performance, we simulate only guide hairs with explicit collision resolution, and then interpolate the states of normal hairs from guide hairs. Our method does not depend critically on any particular hair simulation model as long as the reduced simulation guarantees interactive performance. In our implementation, we use the mass-spring strand dynamic model [Selle et al. 2008] to provide plausible and physically based hair motions, even though it is not originally intended for realtime simulation.

Hair Skinning. To interpolate normal hairs, we propose to exploit precomputed full simulation data to determine interpolation weights. This enables a flexible hair skinning model that can be automatically set up, and thus significantly distinguish it from previous interpolation methods [Chang et al. 2002]. In particular, instead of interpolating an entire strand of a normal hair from guide hairs, we propose to interpolate every normal hair particle from a subset of hair particles on guide hairs (see Figure 3).

Figure 4: Examples of a training sequence *driven by different head motions (shown in individual rows).*

We first define a rest state $\boldsymbol{s}^*$ for all hair particles as the static equilibrium state when the head is static. And the corresponding local coordinate is $\bar{\boldsymbol{s}}^*$. Consider a simulation step, in which the head transformation is $\boldsymbol{T}$ and guide hair particles are moved to their current states $\boldsymbol{s}_g$. For each guide particle g, we first compute a local transformation $\boldsymbol{B}_g$ that transforms it from its rest state $\bar{\boldsymbol{s}}_g^*$ to its current state $\bar{\boldsymbol{s}}_g$ in the local frame, i.e.,

$$\bar{\boldsymbol{s}}_g = \boldsymbol{T}^{-1}\boldsymbol{s}_g = \boldsymbol{B}_g\bar{\boldsymbol{s}}_g^*. \tag{1}$$

When interpolating the state $\boldsymbol{s}_i$ of a normal hair particle i, we consider a set $\mathcal{C}_i$ of guide particles that affect i, and update $\boldsymbol{s}_i$ as

$$\boldsymbol{s}_i = \boldsymbol{T}\left(\sum_{g\in\mathcal{C}_i} w_{ig}\boldsymbol{B}_g\bar{\boldsymbol{s}}_i^*\right), \tag{2}$$

where w_{ig} are positive skinning weights. This skinning model is similar to the classic LBS model widely used in deformable animations. For our purpose, it offers two major advantages. First, it enables us to interpolate normal hairs from guide hairs with fine granularity, as the hair particles along a single normal hair can be affected by different guide hairs. Second, it is easy to train this model from full simulation data. Indeed, this hair skinning model requires three sets of parameters: (i) a set of guide hairs, (ii) a set $\mathcal{C}_i$ of affecting guide particles for every normal particle i and (iii) the associated weights w_{ig}, all of which are determined in our training stage as detailed in the next section.

5 Offline Model Construction

We now present our algorithm to set up the reduced hair model for interactive simulation. In addition, to lay out the foundation of our runtime hair correction algorithm, we will describe our clustering algorithm for creating hair correction groups also based on supplied full simulation data.

5.1 Training Data Preparation

Input to our offline hair simulation includes three parts:

- **A complete hair geometry at rest state** provides a reference state to compute transformation of hair particles in the head's local frame of reference, as used in §4.

Algorithm 1 Guide Hairs Selection

1: Go through all training frames, detect interacting particles pairs and build the topology of graph G
2: Calculate motion similarity of each edge in G according to Eq. (5)
3: Partition G into strand clusters
4: Iteratively select a guide hair in each cluster to maximize energy function (7)

- **A sequence of rigid head motion** is used to drive a full hair animation sequence. They describe the state of the head model at each animation frame, and help to transform hair particle coordinates between the head's local and world frame of reference. We denote them as $T_f, f = 1...F$.
- **A full hair animation sequence** serves as our training data set. Many hair simulation models driven by prescribed head motions can be used to generate the training sequence. In our implementation, we adopt the mass-spring model [Selle et al. 2008] to simulate a full set of hairs with F frames.

To prepare the training data, we select about 8 typical head movements including roll, yaw, pitch and other random motions (see Figure 4). Based on these movements, our full simulation generates an animation with about 500 frames (i.e., $F = 500$).

5.2 Guide Hair Selection

With the training data prepared, we now select a small set of guide hairs for runtime simulation. At first glance, this problem is similar to skinning mesh animations [James and Twigg 2005; Le and Deng 2012]: we need to select guide hairs as bones and skin with normal hairs. However, fundamental differences exist, and therefore direct application of those methods would fail. First, a guide hair is not really a "bone", since it can twist, curl or tangle, and would never be rigid or simply deformed as in conventional LBS models. Consequently, we select individual hair strands as guide hairs, but build the skinning model on individual hair particles instead of entire hair strands. Moreover, we need to handle a large number of hair particles (about 2 millions in our experiments), and select hundreds of guide hairs. The problem size is much larger than a typical mesh skinning problem.

We therefore propose to estimate the hair skinning model in two steps. First, we cluster hair strands into groups using a motion similarity graph, and select a single representative hair from each group as the guide hair. Next, we estimate the skinning weights w_{ig} based on the training sequence (see an outline in Algorithm 1).

Similarity Graph of Strand Motion. We first build a graph model that reflects the local motion similarity of hair strands. We note that Iben et al. [2013] also relies on a graph model built upon the static initial hair geometry to cluster hair-hair contacts. In contrast to their method, our goal is to analyze motion similarity, and thus we consider the entire input animation sequence, in which hairs might move drastically away from their initial states.

We first build a graph on hair particle level instead of strand level, because a single hair may twist or curl differently at its different parts. On the graph, every node represents a hair particle. We initialize graph edges by considering hair particle proximity. Let $\boldsymbol{p}_i(f)$ denote the position of particle i at an input frame f. For every particle i at f, we search its nearby particles within a radius r_l and create an edge between them. In other words, we initialize an edge between i and j as long as there exists a frame f such that $\|\boldsymbol{p}_i(f) - \boldsymbol{p}_j(f)\|_2 < r_l$. The radius r_l is the distance threshold used

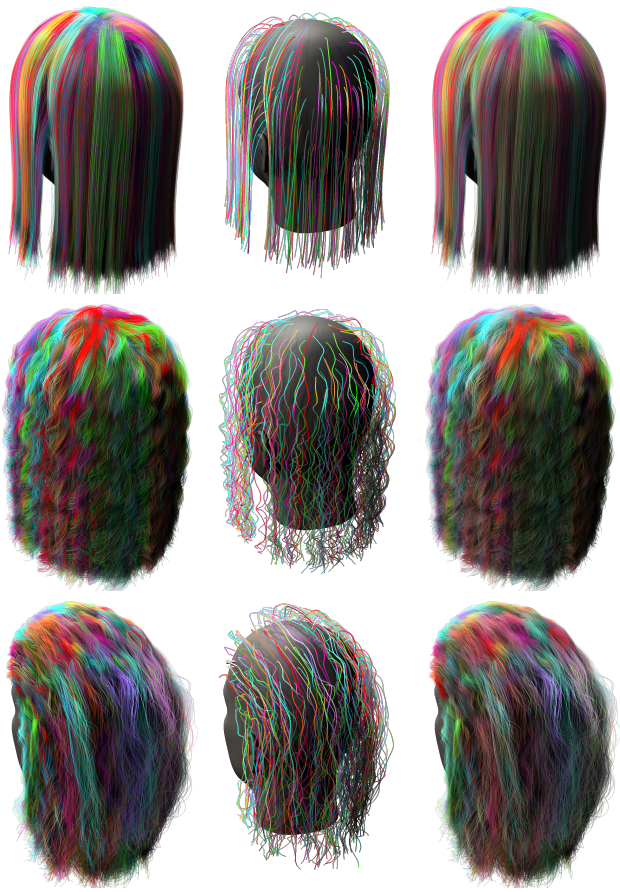


Figure 5: Hair Groups, Guide Strands, and Skinning Weights: *We illustrate the hair groups, guide hairs and skinning weights with different hair styles. (Left Column) The hair particles are clustered into hair groups. (Middle Column) Next, a small number of guide hairs are selected from each group. (Right Column) Lastly, we build a hair skinning model (§5.3) We visualize the hairs with a colormap encoding the skinning weights from guide hair particles.*

in our full simulation to allow hair particle interactions. Meanwhile, for every edge e_{ij}, we maintain an integer number c_{ij} that counts the number of frames in which i and j are in proximity. Namely,

$$c_{ij} = \sum_{f \in F} \delta(\boldsymbol{p}_i(f), \boldsymbol{p}_j(f)), \tag{3}$$

where the indication function $\delta(\boldsymbol{p}_i(f), \boldsymbol{p}_j(f))$ is defined as

$$\delta(\boldsymbol{p}_i(f), \boldsymbol{p}_j(f)) = \begin{cases} 1, & \|\boldsymbol{p}_i(f) - \boldsymbol{p}_j(f)\|_2 \leq r_l \\ 0, & \|\boldsymbol{p}_i(f) - \boldsymbol{p}_j(f)\|_2 > r_l \end{cases}. \tag{4}$$

To enable efficient use of graph algorithms, we further prune the graph edges by removing the edges whose counts are smaller than a threshold, i.e., $c_{ij} < \epsilon_e F$. In practice, we set $\epsilon_e = 0.2$.

After building the basic graph topology, we then assign each edge e_{ij} a weight u_{ij} to measure the motion similarity between hair particle i and j. Consider a particle i at frame f. Let $\bar{\boldsymbol{s}}_i(f)$ and $\bar{\boldsymbol{s}}_i{}^*$ denote its local coordinate of the current state and rest state respectively. We also compute a local transformation $\boldsymbol{B}_i(f)$ that transforms the particle from $\bar{\boldsymbol{s}}_i{}^*$ to $\bar{\boldsymbol{s}}_i(f)$ (i.e., $\bar{\boldsymbol{s}}_i(f) = \boldsymbol{B}_i(f)\bar{\boldsymbol{s}}_i{}^*$). Then the weight u_{ij} is defined as

$$u_{ij} = -\sum_{f \in F} \left(\|\bar{\boldsymbol{s}}_i(f) - \boldsymbol{B}_j(f)\bar{\boldsymbol{s}}_i^*\|_2^2 + \|\bar{\boldsymbol{s}}_j(f) - \boldsymbol{B}_i(f)\bar{\boldsymbol{s}}_j^*\|_2^2 \right) + C_w, \tag{5}$$

where C_w is a constant to ensure u_{ij} is always positive. In practice, we simply use the maximum value of the first term in Eq. (5):

$$C_w = \max_{i,j} \sum_{f \in F} \left(\|\bar{\boldsymbol{s}}_i(f) - \boldsymbol{B}_j(f)\bar{\boldsymbol{s}}_i^*\|_2^2 + \|\bar{\boldsymbol{s}}_j(f) - \boldsymbol{B}_i(f)\bar{\boldsymbol{s}}_j^*\|_2^2 \right).$$

This weight function is invariant to head poses, since we compute the difference in the head's local frame. It further guarantees that a pair of particles with similar motions produces a large weight, which is a property allowing to apply the K-way cut algorithm in the next step.

Before applying our guide hair selection algorithm, we finally shrink the size of the graph. We contract all hair particle nodes that come from a single hair strand into a single graph node. We also sum up the weights of edges that are contracted into a single one, and use the summation as the weight of the resulting edge. After this step, we have a graph whose nodes correspond to individual hair strand. Our next step is to cluster this graph into groups and select a representative node from each group as a guide hair.

Graph Partition. We partition the graph into k parts to group hairs with coherent motions. Formally, we cut the graph into k disjoint sets $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_k\}$ while minimizing

$$f = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{a \in \mathcal{S}_i, b \in \mathcal{S}_j} u_{ab}. \tag{6}$$

This is a classical minimum K-cut problem. When k is user-specified as in our case, it is NP-hard. Instead, we adopt approximated K-cut algorithm [Vazirani 2001], which provides fast performance and guarantees bounded approximation results. In practice, we use the approximated K-cut algorithm provided by the library METIS [Karypis and Kumar 1998].

Selection of Guide Hairs. We now select a guide hair from each group, which consists of coherently moving hairs. Although there are many choices on guide hair selection, given the large size of graph nodes, we prefer a method that balance the computation cost and the quality of the resulting skinning model. Therefore, we propose an hair selection algorithm that maximizes the following energy function

$$f(\mathcal{G}) = \sum_{i \notin \mathcal{G}} \sum_{j \in \mathcal{N}(i) \cap \mathcal{G}} u_{ij}, \tag{7}$$

where $\mathcal{G}$ is the set of selected guide hairs; i denotes a normal hair; and j denotes a guide hair in its neighborhood $\mathcal{N}(i)$ (i.e., there exists an edge e_{ij} on the graph). We then select guide hairs to maximize the similarity between every normal hair and the guide hairs in its neighborhood using an iterative algorithm:

- **Initialization:** We select in each group the node that has the maximum summation of edge weights. Note that these nodes can not guarantee a maximum energy value of Eq. (7), because two selected nodes may connect with each other.
- **Iteration:** We then sequentially operate on every single group and fix other groups. We traverse each node in the current group and compute the energy function value when using that node as the guide hair. We select the node that leads to the maximum energy value as the guide hair, and move on to the next group.
- **Termination:** We repeat the iteration until selected guide hairs $\mathcal{G}$ stop changing. Since at each iteration the energy function never decreases, this algorithm always converges and terminates.

Given a large graph, this heuristic algorithm is fast and produces plausible skinning model configuration. In Figure 6, we compare its

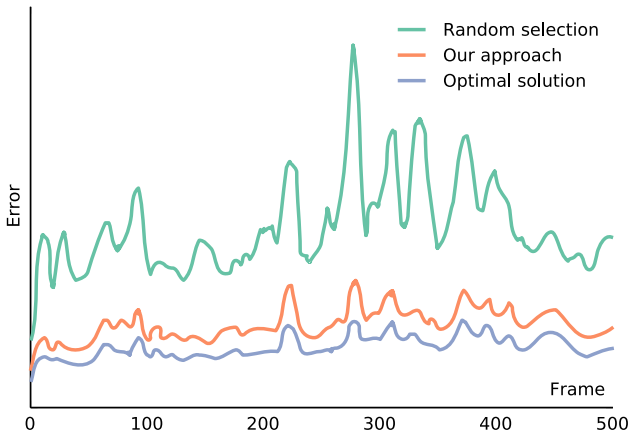
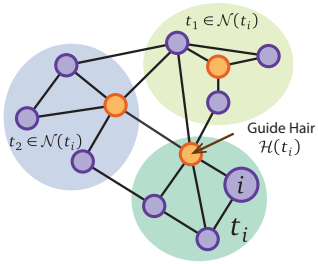


Figure 6: Reconstruction Errors *of different guide hair selection strategies: the error metric is defined as Eq. (8). Randomly selected guide hairs (green curve) lead to large error. A brute-force hair selection strategy (Blue curve) performs iteratively over every group to find the guide strands that lead to the smallest error, taking about 10 hours. Our heuristic approach (orange curve) is slightly less optimal, but much faster, taking less than 10 minutes.*

performance against the results of directly optimizing the reconstruction error of the skinning model, which is mathematically optimal, but computationally much more expensive.

5.3 Estimation of Skinning Weights

After the selection of guide hairs, we finally estimate the skinning weights w_{ig} for every normal hair particle, and complete our skinning model setup. Here for every normal hair particle i, we decide (i) a set $\mathcal{C}_i$ of guide hair particles that affect its position interpolation and (ii) the skinning weight w_{ig} for every guide hair particle $g \in \mathcal{C}_i$ (see Algorithm 2).



We start by introducing a few notations. Consider a normal particle i in one of the graph parts that we partition into in §5.2. Let t_i denote the graph part that contains i and $\mathcal{N}(t_i)$ denote the set of other graph parts connecting with t_i. In other words, for any graph part $t \in \mathcal{N}(t_i)$, there exists an graph edge (a, b) such that $a \in t$ and $b \in t_i$. In addition, we use $\mathcal{H}(t)$ to denote the set of guide hair particles on the guide hair of graph part t. See the adjacent figure for an illustration of these notations.

Initially, given a normal particle i, we choose its guide particle set as $\mathcal{C}_i = \mathcal{H}(t_i) \cup \mathcal{H}(t)\ \forall t \in \mathcal{N}(t_i)$. Then we estimate a skinning weights w_{ig} such that the resulting skinning positions for all normal particles at every input frame are as close as possible to their provided positions in the training sequence. Concretely, we solve a least-square minimization problem similar to [James and Twigg 2005; Wang and Phillips 2002],

$$\min_{w_i} \sum_{f=1}^{F} \| \sum_{g \in \mathcal{C}_i} w_{ig} \boldsymbol{B}_g(f) \bar{\boldsymbol{s}}_i^* - \bar{\boldsymbol{s}}_i(f) \|^2, \tag{8}$$

where, consistent with earlier notations, $\bar{\boldsymbol{s}}_i(f)$ and $\bar{\boldsymbol{s}}_i^*$ are respectively the local coordinates of the current and rest states of particle i; and $\boldsymbol{B}_i(f)$ transforms the particle from $\bar{\boldsymbol{s}}_i^*$ to $\bar{\boldsymbol{s}}_i(f)$. This approximation is linear with respect to the unknowns w_{ig}, which should also satisfy the constraints

$$w_{ig} \geq 0 \ \text{ and } \ \sum_{g \in \mathcal{C}_i} w_{ig} = 1. \tag{9}$$

We simply solve this linearly constrained least-square problem using a quadratic programming solver implemented using QL [Schittkowski 2005].

Algorithm 2 Skinning Weights Estimation

1: **for** each normal hair particle i **do**
2: Find all affecting guide particles C_i
3: Estimate skinning weights W_{ig} for each $g \in C_i$ with Eq. (8)
4: Limit particle size in C_i and re-estimate skinning weights
5: **end for**

At this point, $\mathcal{C}_i$ can have tens of guide particles. A large $\mathcal{C}_i$ size will degrade our runtime skinning performance. Therefore, we further limit its size such that $|\mathcal{C}_i| \leq M$ for every particle i. To this end, we discard guide particles from $\mathcal{C}_i$ and only keep M particles with the largest weight w_{ig}. In practice, we use $M = 10$. This step finalizes the guide particle set $\mathcal{C}_i$. Finally, we use the updated $\mathcal{C}_i$ and solve the constrained least-square problem (8) to finalize the skinning weights.

5.4 Clustering Hair Correction Groups

In addition to setting up the hair skinning model, we cluster all hair particles into a number of groups, called hair correction groups. The purpose of these groups is to accelerate the collision resolution step in the interactive hair simulation. We will present the collision resolution details later in §6.2. For now, we note that our goal here is to partition the hair particles into groups with minimal inter-group collision interactions. At runtime, we resolve collisions inside each group in parallel, and use a few iterations to address the weak interactions across groups. Readers who are not familiar with the hair collision resolution details may skip this section temporarily and refer it back from §6.2.

Again, we address this clustering problem using a graph model. In fact, we reuse the graph that we create at the beginning of §5.2. We briefly review the graph here. In the graph, each node corresponds to a hair particle. An edge e_{ij} is created if there exists a frame f such that $\|\boldsymbol{p}_i(f) - \boldsymbol{p}_j(f)\|_2 < r_l$. Now, the weight of an edge e_{ij} is the number of frames in which i and j interact each other, computed as in Eq. (3).

This graph reflects how frequently two hair particles interact. We seek to cut this graph into K parts, each forming a correction group. Meanwhile, to balance the runtime parallel computation, we hope the size of each group to be as close as possible. Formally, our graph partitioning problem is to partition the graph nodes into K groups with roughly equal sizes such that the total weights of edges connecting nodes in different groups is minimized. This is yet another constrained K-cut problem, and we again use the approximation algorithm implemented in METIS [Karypis and Kumar 1998] to solve it. In all our results, we set $K = 100$ to take advantage of parallelism for runtime collision resolution.

6 Interactive Hair Simulation

With a reduced hair model setup, we proceed to present our interactive hair simulation. In general, our simulation method has two steps: (i) we simulate the selected guide hairs and use them to interpolate a full head of hair; (ii) we correct interpolated hair motions to further resolve detailed collisions. The first step generates

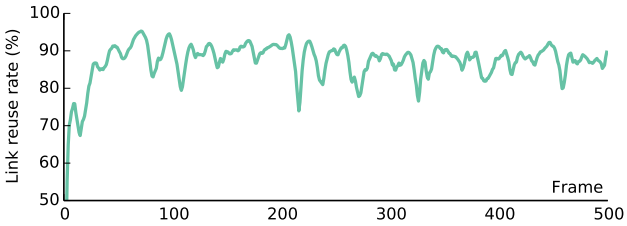


Figure 7: Reuse of Collision Links: *In a typical hair simulation, our temporally coherent link update reuses more than 80% of links from previous frames on average.*

smooth motion in large space scales, while the second step fills in details. However, large-scale interactive simulation allocates very limited time budget for these steps. For instance, we must finish a timestep within 30ms in a practical system with many other components. Therefore, we optimize both steps and exploit parallelism and temporal coherence throughout the simulation.

6.1 Simulation of Reduced Hair Model

At runtime, we simulate only guide hairs. While many hair simulation methods are available, our implementation is largely based on the mass-spring model [Selle et al. 2008] as also used for training data generation. We detect hair-body collision using a level-set representation of solid geometry and process collisions as in [Bridson et al. 2003]. Hair-hair interactions are approximated with a penalty model, which considers both repulsion and stiction effects when two particles are close to each other. At each time step, we create a spring link between two hair particles whose distance is less than a threshold. As they move away from each other, we dynamically remove the spring links. We refer the reader to their paper for implementation details.

Temporally Coherent Link Update. We find that one critical performance bottleneck at every timestep is the spring link update, because the interacting pairs of particles change over time. While various spatial acceleration data structures exist to accelerate the nearest neighbor searches (NNS), it is still expensive to perform at each timestep given limited time budget. To make matters even worse, a sudden change of head motion may push together hair strands, and thus severely increase the number of collision pairs. Such a burst of collision pairs suddenly slows down the simulation and unstabilizes the frame rate, resulting in an unpleasant lag in a realtime system.

To alleviate this problem, we exploit temporal coherence and propose a limited link update algorithm. First, we limit the maximum number (denoted as N) of links that a hair particle can have at a timestep. This enables us to use k-nearest neighbor (kNN) search, which outperforms the fix-radius search because of more efficient pruning. Next, we notice that the link distribution varies temporally coherently, because the distance between two particles always changes gradually over time. Therefore, at the beginning of our collision detection, we first check if a particle's links from previous timestep are still within the penalty distance threshold. We only discard existing links that becomes invalid at the current timestep. If the number (denoted as M) of remaining links is less than N, we search for additional $N-M$ nearest particles, and check if we need to create extra spring links. Because of the temporal coherence of links, $N-M$ tends to be very small, leading to small-scale and thus fast kNN searches. In our experiments, we find that this algorithm reuses more than 80% links on average (see Figure 7), and thus largely improve the runtime performance.

6.2 Hair Correction for Detailed Collision Resolution

After advancing the states of guide hairs at a timestep, we interpolate the states of all normal hairs using the hair skinning model introduced in §4. Not surprisingly, the resulting animation lacks detailed motions, because the hair skinning ignores collisions. Our final step in the interactive simulation is to correct the interpolated hair states to resolve collisions and thus recover hair details.

If the total number of normal hairs is extremely large (e.g., more than 150K hairs), we can extend our model and organize hairs as a three-level structure and perform hair correction only at the middle level with about 10K strands to guarantee realtime performance. We defer the details of this minor extension in §7.

Position-Based Collision Resolution. We now correct all hair particles to resolve collisions. Given a limited time budget and a large number of hair particles, it it necessary to keep the collision resolution as light as possible. Therefore, a natural choice is to correct hair particle positions using position-based dynamics (PBD) [Müller et al. 2007]. Our first attempt was to follow the PBD and project hair particles using collision constraints with a few Gauss-Seidel iterations. Unfortunately, this correction is independent at each timestep, disregarding the temporal coherence of hair particles. It leads to rather pronounced flickering artifacts, because some corrected hairs can move back and forth with respect to other hairs.

Instead, we propose to ensure the temporal coherence using a regularized energy function in PBD framework. In particular, we correct hair particle positions by minimizing the cost function,

$$g = \sum_{i\in\mathcal{P}} \frac{1}{2} m_i \|\boldsymbol{p}_i' - \boldsymbol{p}_i\|_2^2 + \frac{kh^2}{2} \sum_{(i,j)\in\mathcal{L}} \left\| \boldsymbol{p}_i' - \boldsymbol{p}_j' - d_r \frac{\boldsymbol{p}_i - \boldsymbol{p}_j}{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2} \right\|_2^2, \tag{10}$$

where $\boldsymbol{p}_i'$ is the corrected particle position that we need to solve; $\boldsymbol{p}_i$ is the position resulting from hair skinning model; m_i is the mass of particle i; h is the simulation timestep size; d_r is the collision distance threshold as used in our penalty-based collision model; $\mathcal{L}$ indicates the set of particle pairs whose distance is less than d_r. The second term of g penalizes the violation of collision constraints, while the first term penalizes the deviation of correction positions from the interpolated and temporally coherent positions. Lastly, k is the parameter to balance both terms, also used as a stiffness parameter in the PBD framework.

As another justification of the cost function Eq. (10), we note that $\boldsymbol{p}_i$ can be seen as predicted positions. And minimizing Eq. (10) amounts to solving a correction equation, $\mathsf{M}(\boldsymbol{p}' - \boldsymbol{p}) = h^2 \nabla C(\boldsymbol{p}', \boldsymbol{p})$, where M is the mass matrix consisting of m_i, $\boldsymbol{p}$ and $\boldsymbol{p}'$ are vectors stacking the predicted and corrected hair particle positions respectively, and $C(\boldsymbol{p}', \boldsymbol{p})$ is the collision energy function used as the second term in Eq. (10).

We note that similar formulation has been developed in the recent work [Liu et al. 2013]. Here we emphasize the fundamental differences to clarify the reason why we can not apply their method and to motivate our proposed solution in the rest of this section. First, aiming at interactive hair simulation, we have much larger problem size with quite limited computing budget. Even directly solving a sparse linear system as proposed in their method with our problem size is unaffordable in realtime. More importantly, their method assumes a fixed spring connectivity throughout the simulation and thus a constant system matrix that allows pre-factorization for fast solve. Unfortunately, hair simulation breaks this assumption, because hair particles may move close to or away from each other, changing the spring links dynamically. Lastly, facing these difficulties, we choose to use the spring energy function appeared as the second term in Eq. (10). This function distinguishes itself from the conventional spring potential that is used as a critical form to enable

Algorithm 3 Fast Position Correction

Input: Original particle positions P
Output: Corrected particle positions P'

```
for i = 1 → maxIteration do
    for g = 1 → allGroup do
        Correct collision for P' ← p ∈ C_g,
        with boundary conditions p ∉ C_g ← P
    end for
    P ← P'
end for
```

the fast solve in [Liu et al. 2013]. Our energy function instead fixes the spring link direction to linearize the collision forces with respect to $\boldsymbol{p}'_i$, and therefore allows interactive parallel solves.

Fast Solve of Corrected Positions. A direct approach to minimize Eq. (10) is to solve $\nabla_{\boldsymbol{p}'} g = 0$. It solves a sparse linear system because Eq. (10) has a quadratic form. However, given the large number of hair particles, the direct sparse solve is still unaffordable in interactive simulation. For example, in our system, a time budget of 20ms allows only a sparse linear solve with about 2500 hair particles (i.e., about 100 hair strands), much smaller than the desired hair simulation size. We therefore seek for a faster solver. We outline our proposed solver in Algorithm 3, and elucidate its critical components as follows:

- **Jacobi Iterations over Hair Correction Groups:** During the training stage, we cluster hair particles into hair correction groups (see §5.4). The resulting correction groups tend to have weak hair-hair interactions across groups. At the high level, our algorithm takes Jacobi iterations over the correction groups. At each iteration, we resolve the hair particle collisions in the correction groups individually. When operating on one group, we fix the positions of particles in all other groups using the positions from last iteration. The weak coupling among hair correction groups enables a fast Jacobi convergence. In practice, 4 iterations are sufficient to resolve all collisions plausibly. In Figure 8, we justify the advantage of our hair correction groups by comparing its convergence rate with a naive approach.
- **Parallel Solve on Hair Correction Groups:** We use Jacobi iterations over correction groups instead of Gauss-Seidel-type iterations, because it enables us to exploit widespread multiprocessor for solving Eq. (10) in parallel. Consider one correction group with a set $\mathcal{P}_t$ of particles, we solve the corrected particle positions by minimizing the cost function,

$$g_t = \sum_{i\in\mathcal{P}_t} \frac{1}{2} m_i \|\boldsymbol{p}'_i - \boldsymbol{p}_i\|_2^2 + \frac{kh^2}{2} \sum_{\substack{(i,j)\in\mathcal{L} \\ i\in\mathcal{P}_t \text{ or} \\ j\in\mathcal{P}_t}} \left\| \boldsymbol{p}'_i - \boldsymbol{p}'_j - d_r \frac{\boldsymbol{p}_i - \boldsymbol{p}_j}{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2} \right\|_2^2. \quad (11)$$

  Here in the second term, when a particle is linked against another particle j outside of the group, we use the fixed position $\boldsymbol{p}_j$ from last iteration. Given a small number of particles in one correction group, minimizing g_t amounts to solving a small linear system. Thus we can solve it in a short time (usually less than 3ms).
- **Incremental Cholesky Factorization:** Minimizing g_t by solving the equation $\nabla_{\boldsymbol{p}'} g_t = 0$ takes the form $\mathsf{A}\boldsymbol{p}' = \boldsymbol{b}$, in which A is a sparse matrix. Its sparsity pattern depends on particle connectivity. In particular, if the distance between two particles i and j in the group is less then d_r, we add a spring link between them. Correspondingly, we have a rank-3 update of A,

$$\mathsf{A} \leftarrow \mathsf{A} + \boldsymbol{v}\boldsymbol{v}^T, \quad (12)$$

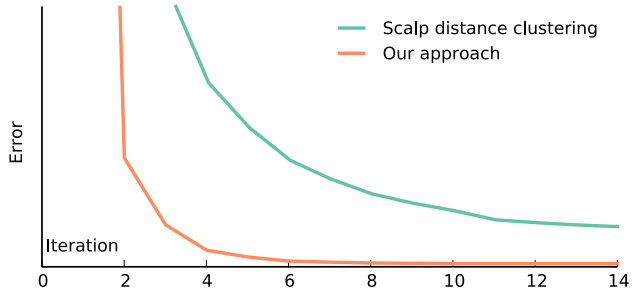


Figure 8: Error convergence curves *of iterative hair correction with different particle clustering strategies: The error is defined as the sum of particle distances to the directly solved optimal particle positions. The green curve is generated using a naive clustering based on geodesic distance of strand roots on head scalp. The orange curve shows the result with our hair correction groups, which requires much fewer iterations to converge.*

where $\boldsymbol{v}^T = \left[0 \ldots kh^2\mathsf{I}_{3\times3} \ldots -kh^2\mathsf{I}_{3\times3} \ldots 0\right]$ is a $3 \times 3n$ matrix, where n is the number of particles in the group; it has zero everywhere except two nonzero 3×3 blocks, $kh^2\mathsf{I}_{3\times3}$ and $-kh^2\mathsf{I}_{3\times3}$, at positions corresponding to particle i and j. Here $\mathsf{I}_{3\times3}$ is a 3×3 identify matrix. Similarly, when a link between i and j is removed, we have another rank-3 update, $\mathsf{A} \leftarrow \mathsf{A} - \boldsymbol{v}\boldsymbol{v}^T$. This low-rank update of A suggests that we once again exploit the temporal coherence of link update. At the beginning of the hair correction step, we use the low-rank incremental Cholesky algorithm [Seeger 2007] to update the Cholesky factorization of A, and reuse it during our Jacobi iterations.

Lastly, our Jacobi iterations always converges to the hair particle positions that minimize Eq. (10), because Eq. (10) is a quadratic form with a unique global minimum, and every hair correction solve in each group always decreases its value.

7 Implementation Details

Hair Simulation. We use a mass-spring model similar to [Selle et al. 2008] for both offline and runtime simulations. We sample every hair strand uniformly in arc-length and build particle spring connections. The number of particles per strand is a critical parameter that affects both the result quality and performance. We found a size of 20-25 in our experiments can achieve a good balance. For different hair styles, we adjust the simulation parameters (e.g., spring stiffness, damping and friction coefficients) to get plausible dynamic behavior. But both offline full simulation and runtime interactive simulation use the same set of parameters, and no further parameter tuning is needed.

Multi-Level Reduced Model. For extremely large number of hairs, our reduced model can be easily extended to perform multi-level interpolation to guarantee the performance. The normal hairs at a coarse level serve as guide hairs for its succeeding fine level. For instance, to ensure realtime performance on a commodity desktop, our current implementation for the most complex examples organizes more than 150K hairs as a three-level structure. The top level consists of a few hundred (200-400 in our experiments) guide hairs simulated directly at runtime. The second level are sparser normal hairs interpolated using guide hairs. And the finest level are the full hairs further interpolated from the second level. We estimate two skinning models for hair interpolation from the first level to the second level and from the second level to the third level. But we perform strand interaction correction only at the second level, which typically consists of about 20K in our experiments.

Hair-Body Collision Correction. While hair-body collisions are

Figure 9: *From left to right: the results of our reduced model without interaction correction, the results of our reduced model with interaction correction, and the full simulation results.*

largely resolved during guide hair simulation, the interpolated normal strands may still penetrate into the body. To efficiently correct hair-body collisions at runtime, we assume both the head and body undergo rigid transformation, and precompute a distance field and the distance gradient (for normals) stored in a level-set lattice. In the correction step, after interpolating all normal hairs, we detect colliding particles by evaluating their distance value from the body surface. All colliding particles are pushed outwards along the precomputed normals using repulsion impulses. To ensure spatial and temporal coherence, we further filter correction impulse vectors by a smooth kernel.

Hair Rendering. We render all figures and animations in this paper using a realtime renderer that uses the single scattering formulation [Marschner et al. 2003] and handles shadows as in [Lokovic and Veach 2000].

8 Experimental Results

We validate our reduced model on a variety of hairstyles, including straight, wavy and curly hairs. Experiments show that the model can generate detailed hair motions and produce similar results as captured by a full simulation. Figure 12 shows some animation results driven by head/body motions (the first three rows) and wind forces (the bottom row).

We have extensively experimented our method on new head motions that are not included in training motions (see Figure 12). Results show that our method generalizes to new head motions well as long as (i) the runtime head rotation is within the range of the training data, and (ii) the head does not move with much larger velocity than that in the training data. We also observed that as long as sufficient head motion variance is included, specific motion in the training data would not greatly affect the resulting reduced model, strand partition or skinning weight distribution.

As shown in Figure 11, we integrated our hair simulation algorithm into a video-based facial tracking system [Cao et al. 2013]. The rigid head transformation tracked by the system is used to drive the hair animation, which is then rendered together with the digital avatar. We are able to run the entire system in realtime. Please see the supplemental video for a live demo. This clearly demonstrates the potential of our reduce hair model in interactive applications.

Performance. We tested the performance of our method on a commodity PC with a quad-core Intel i7-3770K CPU, 32GB memory and Nvidia GeForce GTX 760 graphics card. For a hair model

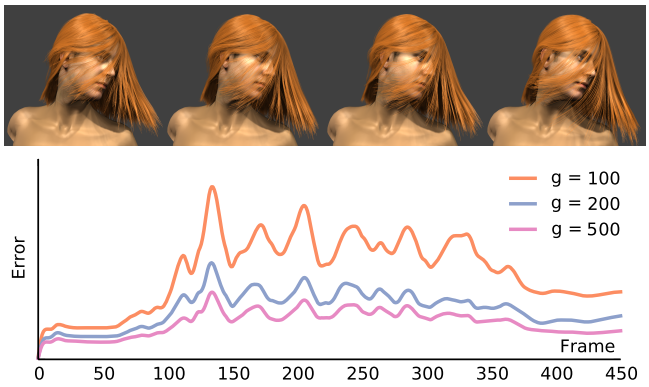


Figure 10: Effects of number of guide hairs*: (Top) one resulting frame simulated using 100/200/500 guide hairs and a full simulation. (Bottom) the respective reconstruction error curves of a training sequence, evaluated according to Eq. (8).*

Figure 11: *We apply our interactive hair simulation to a realtime performance-driven digital avatar.*

with 150K strands and 300 guide hairs, the total simulation time is about 40ms per frame, with 15ms for guide hair simulation, 5ms for interpolation, and 20ms for collision correction.

Our implementation of offline full simulation typically costs 0.5-1 minute per frame for 100K-150K strands. The total simulation time for a training sequence of 500 frames is around 5 hours. During reduced model construction, guide hair selection takes about 10 minutes, and skinning weight estimation takes about 30 minutes.

Comparisons. In Figure 9, we compare results of our reduced model (with and without collision correction) and the full simulation results, under the same head motions that do not exist in the training data. As shown, the collision correction enhances the interpolation results and recover more motion details and wisp structures, resulting in more realistic animations. Please refer to the video for more comparisons. Although one can still see some differences between our results and the full simulation results, they are visually similar with motion details largely preserved. More importantly, our method generates the results in realtime, being over two orders of magnitude faster than the full simulation.

The most important parameter of our reduced model is the number of guide hairs. More guide hairs leads to larger degrees of freedom (DoFs) for runtime simulation, but better interpolation for the full hair model. In practice, we use 200-300 guide hairs in all our results. In Figure 10, we compare the results and reconstruction errors with 100, 200, and 500 guide strands, and we found that guide strands of more than 200 would not significantly improve the visual realism of the final results.

Figure 12: *Animation results generated by our method: Top three rows demonstrate different hair styles, while the bottom row shows the animation driven by wind forces.*

9 Conclusion

We have introduced a data-driven approach to simulating a full head of hairs in realtime, while still preserving similar motion details as in an offline full simulation. Our approach is composed of an optimized hair skinning model with carefully selected guide hairs and interpolation weights, and a position-based correction model that can corrects the interpolated hair motions to respect collisions. We hope our approach is practically useful for many interactive applications such as games and virtual reality systems.

There exists a number of limitations of our algorithm: (i) Currently the same simulation parameters are used for both offline and runtime simulation to produce consistent motion. But our reduced model may not fit well with different parameters or hair geometry. (ii) Our runtime hair correction adds plausible local motion details using a penalty model to resolve collisions, but cannot completely prevent strand mutual collision. Strand penetration can still occur. (iii) Our current strand mutual interaction model lacks realistic friction effects, a more advanced interaction model [Daviet et al. 2011] might be incorporated to further improve the results. (iv) And runtime head motions that are significantly beyond the scope of the training data (as discussed in §8) may produce unsatisfactory results.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback, and Cem Yuksel for sharing his hair models (http://www.cemyuksel.com/research/hairmodels). This research was supported in part by NSFC (No. 61272305), the National High-tech R&D Program (No. 2012AA011503), National Program for Special Support of Eminent Professionals of China, Columbia Junior Faculty Startup Fund as well as generous gifts from Intel. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies or others.

References

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. 27*, 5 (Dec.), 165:1–165:10.

ANJYO, K.-I., USAMI, Y., AND KURIHARA, T. 1992. A simple method for extracting the natural beauty of hair. In *SIGGRAPH Computer Graphics*, vol. 26, 111–120.

BANDO, Y., CHEN, B.-Y., AND NISHITA, T. 2003. Animating hair with loosely connected particles. *Computer Graphics Forum 22*, 3, 411–418.

BERGOU, M., WARDETZKY, M., ROBINSON, S., AUDOLY, B., AND GRINSPUN, E. 2008. Discrete elastic rods. *ACM Trans. Graph. 27*, 3, 63.

BERTAILS, F., KIM, T.-Y., CANI, M.-P., AND NEUMANN, U. 2003. Adaptive wisp tree: a multiresolution control structure for simulating dynamic clustering in hair motion. In *Proceedings of SCA*, 207–213.

BERTAILS, F., AUDOLY, B., CANI, M.-P., QUERLEUX, B., LEROY, F., AND LÉVÊQUE, J.-L. 2006. Superhelices for predicting the dynamics of natural hair. *ACM Trans. Graph. 25*, 3, 1180–1187.

CAO, C., WENG, Y., LIN, S., AND ZHOU, K. 2013. 3D shape regression for real-time facial animation. *ACM Trans. Graph. 32*, 4 (July), 41:1–41:10.

CASATI, R., BERTAILS-DESCOUBES, F., ET AL. 2013. Super space clothoids. *ACM Trans. Graph. 32*, 4, 48.

CHANG, J. T., JIN, J., AND YU, Y. 2002. A practical model for hair mutual interactions. In *Proceedings of SCA*, 73–80.

CHOE, B., CHOI, M. G., AND KO, H.-S. 2005. Simulating complex hair with robust collision handling. In *Proceedings of SCA*, 153–160.

DAVIET, G., BERTAILS-DESCOUBES, F., AND BOISSIEUX, L. 2011. A hybrid iterative solver for robustly capturing Coulomb friction in hair dynamics. *ACM Trans. Graph. 30*, 6, 139.

FENG, W.-W., YU, Y., AND KIM, B.-U. 2010. A deformation transformer for real-time cloth animation. *ACM Trans. Graph. 29*, 4, 108.

GUAN, P., SIGAL, L., REZNITSKAYA, V., AND HODGINS, J. K. 2012. Multi-linear data-driven dynamic hair model with efficient hair-body collision handling. In *Proceedings of SCA*, 295–304.

HADAP, S., AND MAGNENAT-THALMANN, N. 2001. Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3, 329–338.

IBEN, H., MEYER, M., PETROVIC, L., SOARES, O., ANDERSON, J., AND WITKIN, A. 2013. Artistic simulation of curly hair. In *Proceedings of SCA*, 63–71.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph. 22*, 3, 879–887.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Trans. Graph. 24*, 3, 399–407.

KARYPIS, G., AND KUMAR, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing 20*, 1, 359–392.

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A. W., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*, 4, 93.

LE, B. H., AND DENG, Z. 2012. Smooth skinning decomposition with rigid bones. *ACM Trans. Graph. 31*, 6, 199.

LIU, T., BARGTEIL, A. W., O'BRIEN, J. F., AND KAVAN, L. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph. 32*, 6, 214.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *Proceedings of SIGGRAPH*, 385–392.

MARSCHNER, S. R., JENSEN, H. W., CAMMARANO, M., WORLEY, S., AND HANRAHAN, P. 2003. Light scattering from human hair fibers. *ACM Trans. Graph. 22*, 3, 780–791.

MCADAMS, A., SELLE, A., WARD, K., SIFAKIS, E., AND TERAN, J. 2009. Detail preserving continuum simulation of straight hair. *ACM Trans. Graph. 28*, 3, 62.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2, 109–118.

MÜLLER, M., KIM, T.-Y., AND CHENTANEZ, N. 2012. Fast simulation of inextensible hair and fur. In *Workshop on Virtual Reality Interaction and Physical Simulation*, 39–44.

PETROVIC, L., HENNE, M., AND ANDERSON, J. 2005. Volumetric methods for simulation and rendering of hair. *Pixar Animation Studios*.

PLANTE, E., CANI, M.-P., AND POULIN, P. 2001. A layered wisp model for simulating interactions inside long hair. In *Computer Animation and Simulation*. Springer, 139–148.

ROSENBLUM, R. E., CARLSON, W. E., AND TRIPP, E. 1991. Simulating the structure and dynamics of human hair: modelling, rendering and animation. *The Journal of Visualization and Computer Animation 2*, 4, 141–148.

SCHITTKOWSKI, K. 2005. QL: A Fortran code for convex quadratic programming-user's guide, Version 2.11. *Report, Department of Mathematics, University of Bayreuth*.

SEEGER, M. 2007. Low rank updates for the Cholesky decomposition. *University of California at Berkeley, Tech. Rep*.

SELLE, A., LENTINE, M., AND FEDKIW, R. 2008. A mass spring model for hair simulation. *ACM Trans. Graph. 27*, 3, 64.

TARIQ, S., AND BAVOIL, L. 2008. Real time hair simulation and rendering on the GPU. In *ACM SIGGRAPH 2008 talks*, 37.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Trans. Graph. 25*, 3, 826–834.

VAZIRANI, V. V. 2001. *Approximation Algorithms*. Springer.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of SCA*, 129–138.

WANG, H., O'BRIEN, J. F., AND RAMAMOORTHI, R. 2011. Data-driven elastic models for cloth: modeling and measurement. *ACM Trans. Graph. 30*, 4, 71.

WARD, K., AND LIN, M. C. 2003. Adaptive grouping and subdivision for simulating hair dynamics. In *Proceedings of Pacific Graphics*, 234–243.

WARD, K., LIN, M. C., JOOHI, L., FISHER, S., AND MACRI, D. 2003. Modeling hair using level-of-detail representations. In *Proceedings of Computer Animation and Social Agents*, 41–47.

WARD, K., BERTAILS, F., KIM, T.-Y., MARSCHNER, S. R., CANI, M.-P., AND LIN, M. C. 2007. A survey on hair modeling: styling, simulation, and rendering. *IEEE TVCG 13*, 2, 213–234.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Trans. Graph. 28*, 3, 39.