Real-Time Physically Guided Hair Interpolation

JERRY HSU*, University of Utah & LightSpeed Studios, USA
TONGTONG WANG, LightSpeed Studios, China
ZHERONG PAN, LightSpeed Studios, USA
XIFENG GAO, LightSpeed Studios, USA
CEM YUKSEL, University of Utah & Roblox, USA
KUI WU, LightSpeed Studios, USA

(a) *Simulated guide hairs* **(b)** *Conventional interpolated hairs* **(c)** ***Our interpolated hairs***

Fig. 1. *Interpolated rendered hairs from the given guide hairs. (a) the hair model with* 128 *guide hairs; (b)* 106*K rendered hairs with* 1.7*M vertices interpolated from the guide hairs using Linear Hair Skinning (LHS) taking 0.28 ms per frame; (c) rendered hairs interpolated using our method taking 0.34 ms per frame. This is especially challenging for interpolation as almost no two guide hairs are the same shape.*

Strand-based hair simulations have recently become increasingly popular for a range of real-time applications. However, accurately simulating the full number of hair strands remains challenging. A commonly employed technique involves simulating a subset of guide hairs to capture the overall behavior of the hairstyle. Details are then enriched by interpolation using linear skinning. Hair interpolation enables fast real-time simulations but frequently leads to various artifacts during runtime. As the skinning weights are often pre-computed, substantial variations between the initial and deformed shapes of the hair can cause severe deviations in fine hair geometry. Straight hairs may become kinked, and curly hairs may become zigzags.

This work introduces a novel physical-driven hair interpolation scheme that utilizes existing simulated guide hair data. Instead of directly operating on positions, we interpolate the internal forces from the guide hairs before efficiently reconstructing the rendered hairs based on their material model. We formulate our problem as a constraint satisfaction problem for which we present an efficient solution. Further practical considerations are addressed using regularization terms that regulate penetration avoidance and drift correction. We have tested various hairstyles to illustrate that our approach can generate visually plausible rendered hairs with only a few guide hairs and minimal computational overhead, amounting to only about 20% of conventional linear hair interpolation. This efficiency underscores the practical viability of our method for real-time applications.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Procedural animation**.

Additional Key Words and Phrases: Hair Interpolation, Cosserat Rod

ACM Reference Format:
Jerry Hsu, Tongtong Wang, Zherong Pan, Xifeng Gao, Cem Yuksel, and Kui Wu. 2024. Real-Time Physically Guided Hair Interpolation. *ACM Trans. Graph.* 43, 4, Article 95 (July 2024), 11 pages. https://doi.org/10.1145/3658176

1 INTRODUCTION

Hair simulations contribute significantly to believable characters and immersive environments in film, video games, and virtual reality. However, achieving efficient and realistic hair simulations remains an open problem because, on average, a human scalp has about $100k$ - $150k$ hair follicles, and simulating, storing, and transferring complex geometries at such scales poses significant challenges to both computational efficiency and robustness.

*Part of this work was done when Jerry Hsu was an intern at LightSpeed Studios.

Authors' addresses: Jerry Hsu, jerry060599@gmail.com, University of Utah & LightSpeed Studios, Salt lake city, UT, USA; Tongtong Wang, tongttwang@tencent.com, LightSpeed Studios, Shenzhen, Guangzhou, China; Zherong Pan, zrpan@global.tencent.com, LightSpeed Studios, Seattle, WA, USA; Xifeng Gao, xifgao@global.tencent.com, LightSpeed Studios, Seattle, WA, USA; Cem Yuksel, cem@cemyuksel.com, University of Utah & Roblox, Salt lake city, UT, USA; Kui Wu, kwwu@global.tencent.com, LightSpeed Studios, Los Angeles, CA, USA.

0730-0301/2024/7-ART95 $15.00
https://doi.org/10.1145/3658176

A widely used solution is to pre-select a fraction of hair strands as *guide hairs* and only simulate those hair strands [Bertails et al. 2008; Yuksel and Tariq 2010]. Subsequently, all remaining hair strands, referred to as *rendered (or interpolated) hairs*, are synthesized from the guide hairs via interpolation. Yet, conventional methods [Chai et al. 2014; Epic 2024; Rosu et al. 2022; Yuksel et al. 2009] rely solely on the geometric information of guide hairs to determine the shape of rendered hairs. Unfortunately, this approach is prone to severe visual artifacts when guide hairs undergo large deformations, which is not uncommon. An example of this is shown in Fig. 1b, where rendered hairs are unnaturally stretched due to geometric differences of their interpolated guide hairs. Though guide hairs can often adequately represent large-scale hair motion, the resulting rendered hairs may be highly unrealistic when generated through geometric interpolation alone.

In this paper, we propose a novel hair interpolation scheme that utilizes physical information directly obtained from the simulation. Rather than directly interpolating positions, we operate on internal forces by interpolating them from guide hairs and then solve for rendered hair shapes that could reproduce these forces. Aiming for real-time applications, we develop a fast Gauss-Seidel solver that can converge with very few iterations. We introduce an efficient drift correction scheme to stabilize solutions and penalty energies to avoid penetrations between rendered hairs and other geometry (such as the face and shoulders). Our approach can entirely avoid visual artifacts caused by conventional geometric interpolation and reproduce individual rendered hair motion using distinct physical properties. Notably, our method does not require any precomputation or training. We use only readily available runtime simulation data from guide hairs, and our interpolation can be computed independently for each rendered strand with only a minor computational overhead. We have tested various hairstyles to illustrate that our approach can generate visually plausible rendered hairs with only 20% additional computation time over conventional linear hair interpolation. This efficiency underscores the practical value of our method for real-time applications.

2 RELATED WORK

In this section, we briefly summarize prior work on hair simulation and hair interpolation, followed by the interpolation methods with reconstruction.

2.1 Hair Simulation

Hair simulation is computationally expensive due to the large number of hair strands. Over the years, various reduced representations of hair bundles have been proposed to achieve cost-effective approximations. These representations encompass 2D strips [Koh and Huang 2001], cubic lattice structures [Volino and Magnenat-Thalmann 2006], short hair strips [Guang and Zhiyong 2002], and volumetric representations [Lee et al. 2019; Wu and Yuksel 2016]. These representations are transformed into full hairs using baked textures or procedural functions during rendering.

Strand-based representations have recently gained prevalence over card/polygon-based representations due to their superior ability to capture high-fidelity hair details. In earlier works, hair strands have often been modeled utilizing techniques such as mass-spring chains [Rosenblum et al. 1991], rigid multi-body chains [Anjyo et al. 1992; Chang et al. 2002], and the incorporation of ghost particles [Umetani et al. 2015] or altitude springs [Selle et al. 2008] to address the twisting effects, particularly pertinent in simulating curly hair. Several elastic rod models have been introduced to achieve a physically accurate representation of hair strands. These models include the super-helix model [Bertails et al. 2006], Cosserat rod elements (CoRdE) [Spillmann and Teschner 2007], discrete elastic rods (DER) [Bergou et al. 2008], damped exponential time integrator (DETI) model [Michels et al. 2015], and position-based Cosserat rods [Kugelstadt and Schömer 2016]. Recently, several works have been introduced to accelerate the strand-based hair simulation scheme, e.g., resolving collisions on a background Eulerian grid [Fei et al. 2021; Han et al. 2019; Hsu et al. 2023; Huang et al. 2023; McAdams et al. 2009] and accelerating the computation using ADMM solver [Daviet 2020, 2023].

2.2 Hair Interpolation

To address the computational demands associated with simulating individual hair strands, a common strategy is to simulate a limited number of guiding hair strands before employing interpolation to reconstruct the dense hair model. Currently, the predominant approach involves the manual authoring strand-based hairstyles, with hair interpolation achieved through Linear Blend Skinning, also known as Linear Hair Skinning (LHS). LHS is extensively used in both gaming [Epic 2024] and film production [Somasundaram 2015]. Unfortunately, LHS may introduce undesirable artifacts, such as zigzagging shapes and shortened/stretched strands, especially when the associated guide hairs exhibit disparate shapes and motions (see Fig. 1b), which only becomes more prominent the more complex the hairstyle, particularly with those involving curly hair.

As these artifacts are often the result of poorly selected LHS weights, several approaches have been proposed to address weight computation. Chai et al. [2014] proposed a data-driven solution reliant on full hair simulation as training data. However, its efficacy heavily depends on having a similar range and motion between full simulation at training and guide simulation at runtime. Since these simulations often do not match behavior, ground truth can be challenging to obtain, limiting the generalization of data-driven methods. As a result, subsequent data-driven approaches [Chai et al. 2017; Lyu et al. 2022] still face challenges related to the need for extensive training and potential artifacts when guide hairs deviate from the training data. Chai et al. [2017] proposed the dynamic selection of weights at runtime to allow for local weight adaptations. However, this approach only considers large-scale motion similarity and fails to account for any runtime fine geometry deviations (e.g., curls) between the guide and interpolated hairs. This is especially true with sparse guide hairs that each has drastically different shapes, as shown in Fig. 1b.

To further account for changes in geometry, Lyu et al. [2022] introduced a neural interpolator for dynamic weight prediction, addressing previous generalization issues at the cost of computation time. In the same vain, Rosu et al. [2022] employed a pre-trained Variational Autoencoder (VAE) to convert the shape of guide hairs

into latent vectors, generating full hairs by decoding these interpolated latent vectors. Strand-VAE [Shen et al. 2023; Sklyarova et al. 2023] has extended this to reconstructing full hairs from a limited number of hair strands. However, the long inference times associated with this method hinder its real-time application, and its outcomes are constrained by the limitations of the training dataset, particularly concerning curly hair.

2.3 Interpolation Methods with Reconstruction

Our method is conceptually similar to the class of techniques known as gradient-domain reconstruction [Huang et al. 2006, 2010; Levin et al. 2004; Zheng 2013]. These methods apply various modifications to the gradient field of images and meshes and then reconstruct the images or meshes by solving a linear system, during which the spurious noise in the gradient domain is eliminated. We observe that the gradient field in a physical simulation corresponds exactly to the forces derived from physical strains and, thus, can be trivially obtained from existing simulations. As such, we propose an interpolation that instead operates on material-aware internal forces. This retains the benefits of gradient-domain reconstruction while making our method aware of physical properties like differing rest shapes and materials.

Physical material properties have been known to integrate well with physical systems. Specifically, approaches based on *differential coordinates* often utilize material-strain-based mesh reconstruction techniques reminiscent of gradient-domain reconstruction. For example, guided simulation methods [Barbič et al. 2009; Schumacher et al. 2012] define penalties based on differences in material strains to match target keyframes or poses in simulation. Strain-based interpolation methods for shells [Fröhlich and Botsch 2011; Heeren et al. 2014; Winkler et al. 2010] minimize differences in strains to produce interpolated poses. However, there are several important distinctions from our technique. Specifically, our method is designed to interpolate deformations between many inherently different hairs instead of the same mesh but with different poses. As such, we do not obtain strains from pre-made poses but rather directly from a running simulation. Furthermore, instead of operating in seconds, our solver is designed to operate in microseconds without the expensive and complex process in computing geodesic paths [Heeren et al. 2014], for example. Our solver is custom-tailored to the non-linear hair reconstruction problem that addresses the practical concerns for stability and performance.

3 STRAND-BASED HAIR SIMULATION

We employ Cosserat Rods to simulate hair dynamics due to their capability to capture complex behavior with an efficient position-based dynamics (PBD) solver [Kugelstadt and Schömer 2016], alongside their widespread adoption in modern game engines, e.g., Unreal Engine Groom [Epic 2024]. Here, we provide a very brief overview of our guide hair simulation framework.

Cosserat theory considers stretching, shearing, twisting, and bending resistance along the rod. Under the discretized representation, each strand segment is attached with a frame, denoted as a quaternion $\mathbf{q}_j$, with two ending vertices at $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$. We use the subscript $_j$ to index an ordered vertex or segment on each hair.

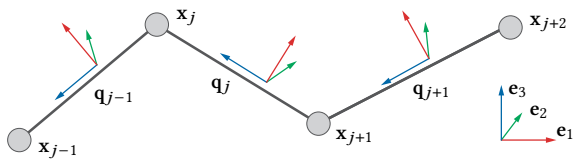

Fig. 2. *Discretized Cosserat rod with position* $\mathbf{x}$ *on vertices and orientation* $\mathbf{q}$ *on segments.* $\mathbf{e}_\alpha$ *is the axis-aligned orthonormal basis*

For each reference frame, we denote the αth axis as a 3D vector $\mathbf{d}_{j,\alpha} = \mathbf{q}_j \mathbf{e}_\alpha \overline{\mathbf{q}}_j$, where $\overline{\mathbf{q}}$ indicates quaternion conjugation and 3D vector $\mathbf{e}_\alpha$ is the αth axis-aligned basis. By convention, we use $\alpha = 3$ as the tangent axis along the segments of hair strands. The arrangements of variables are illustrated in Fig. 2.

Under this notation, Cosserat Rod formulates the rod material with two types of constraints: the *stretching and shearing* constraints $\mathbf{C}^{\text{ss}}$ and the *bending and twisting* constraints $\mathbf{C}^{\text{bt}}$. $\mathbf{C}^{\text{ss}}$ penalizes stretching with respect to the rest length l_j and shearing from the tangent direction as:

$$\mathbf{C}^{\text{ss}}_j \triangleq \mathbf{C}^{\text{ss}}(\mathbf{x}_j, \mathbf{x}_{j+1}, \mathbf{q}_j) = \frac{1}{l_j}(\mathbf{x}_{j+1} - \mathbf{x}_j) - \mathbf{d}_{j,3}. \tag{1}$$

To model the bending and twisting behaviors, $\mathbf{C}^{\text{bt}}_j$ is introduced to penalize the difference between the scaled Darboux vector at current and rest pose as:

$$\mathbf{C}^{\text{bt}}_j \triangleq \mathbf{C}^{\text{bt}}_j(\mathbf{q}_j, \mathbf{q}_{j+1}) = \overline{\mathbf{q}}_i \mathbf{q}_{j+1} - \phi \mathbf{q}^0_j, \tag{2}$$

$$\phi \triangleq \begin{cases} +1 & \text{for } \|\overline{\mathbf{q}}_j \mathbf{q}_{j+1} - \mathbf{q}^0_j\|^2 \le \|\overline{\mathbf{q}}_j \mathbf{q}_{j+1} + \mathbf{q}^0_j\|^2 \\ -1 & \text{for } \|\overline{\mathbf{q}}_j \mathbf{q}_{j+1} - \mathbf{q}^0_j\|^2 > \|\overline{\mathbf{q}}_j \mathbf{q}_{j+1} + \mathbf{q}^0_{ji}\|^2 \end{cases}. \tag{3}$$

where quaternion $\mathbf{q}^0_j = \overline{\mathbf{q}}^0_j \mathbf{q}^0_{j+1}$ represents the scaled Darboux vector at the rest pose. Note that further scaling the stiffness k^{bt} by l_j^{-2} makes our $\mathbf{C}^{\text{bt}}_j$ equivalent to that used by Kugelstadt and Schömer [2016] with discrete Darboux vectors. The energy potential E is further specified in terms of constraint functions $E^\star_j = \frac{1}{2} k^\star \mathbf{C}^{\star T}_s \mathbf{C}^\star_j$, where $\star$ indicates constraint type and $k^\star$ indicates the corresponding stiffness. Finally, we obtain the force and torque due to $\mathbf{C}^{\text{ss}}_j$ on consecutive vertices j and $j+1$ as follows:

$$\mathbf{f}^{\text{ss}}_{j,j} = \frac{k^{\text{ss}}}{l_j}\left(\frac{\mathbf{x}_{j+1} - \mathbf{x}_i}{l_j} - \mathbf{d}_{j,3}\right), \tag{4}$$

$$\mathbf{f}^{\text{ss}}_{j,j+1} = -\frac{k^{\text{ss}}}{l_i}\left(\frac{\mathbf{x}_{j+1} - \mathbf{x}_i}{l_j} - \mathbf{d}_{j,3}\right), \tag{5}$$

$$\boldsymbol{\tau}^{\text{ss}}_{i,i} = -2k^{\text{ss}}\left(\frac{\mathbf{x}_{j+1} - \mathbf{x}_i}{l_j} - \mathbf{d}_{j,3}\right)\mathbf{q}_i \mathbf{e}_3. \tag{6}$$

Similarly, the torque from $\mathbf{C}^{\text{bt}}_j$ on consecutive segments j and $j+1$ reads:

$$\boldsymbol{\tau}^{\text{bt}}_{j,j} = -k^{\text{bt}}\left(\mathbf{q}_j - \phi \mathbf{q}_{j+1}\overline{\mathbf{q}}^0_j\right), \tag{7}$$

$$\boldsymbol{\tau}^{\text{bt}}_{j,j+1} = -k^{\text{bt}}\left(\mathbf{q}_{j+1} - \phi \mathbf{q}_j \mathbf{q}^0_j\right). \tag{8}$$

The extended position-based dynamics (XPBD) time integrator [Macklin et al. 2016] is used to apply the forces and torques to update each vertex's orientation $\mathbf{q}_j$, angular velocity $\boldsymbol{\omega}_j$, position

$\mathbf{x}_j$, and velocity $\mathbf{v}_j$. At the end of each iteration, the quaternion is normalized. We refer readers to the previous work [Kugelstadt and Schömer 2016] for more details on the XPBD integration of Cosserat rods. Then, to handle collisions, hair vertices are coupled with material point particles on an auxiliary Eulerian grid [Fei et al. 2021; Hsu et al. 2023; Huang et al. 2023]. For additional implementation details, we follow and refer readers to Hsu et al. [2023].

4 PHYSICALLY GUIDED STRAND RECONSTRUCTION

In this section, we introduce our novel hair interpolation scheme utilizing real-time hair simulation data to reconstruct visually plausible rendered hair shapes.

To motivate our method, consider the simple case of two hair strands under gravity. No matter their shapes, the strands will exhibit similar internal forces in resistance to the same gravity field. Notably, the forces themselves do not specify a particular hair shape. So, as long as we obtain similar internal forces from nearby guides, the rendered hairs should be able to deform naturally according to their rest shapes. Hence, internal forces should be interpolated before reconstructing positions. We call this process as *force-based position reconstruction*.

Though intuitive, performing such an interpolation robustly and efficiently is challenging. Unlike conventional gradient-domain shape reconstruction [Huang et al. 2006; Levin et al. 2004], which amounts to solving a linear system, the position of a Cosserat rod is related non-linearly to forces due to the incorporation of the quaternion $\mathbf{q}_i$. Therefore, the position is typically solved using a non-linear global optimization [Winkler et al. 2010]. However, these optimizations cannot be performed at real-time in our hair interpolation scenario.

From another perspective, there exists a natural correspondence between conventional rod models and differential coordinates approaches in force-based position reconstruction. Since most rod models [Bergou et al. 2008; Bertails et al. 2006; Kugelstadt and Schömer 2016] consider quadratic potentials with respect to strain, the potential energy derivative (i.e., force) is a simple scalar multiple of the strain. As such, our method can also be interpreted as interpolating strains prior to position reconstruction.

In the following, we begin with Sec. 4.1 by formulating our constraint-based shape reconstruction scheme, which computes positions and orientations that generate our desired internal forces. We further propose several techniques to improve the robustness and visual quality of our solver, including penetration resolution (Sec. 4.2), drift correction (Sec. 4.3), and warm-starting (Sec. 4.4).

4.1 Force-based Reconstruction

Our method requires an existing method to compute interpolated forces on each rendered hair. To this end, we employ Conventional Linear Hair Skinning (LHS). We use the double subscript $_{k,j}$ to denote the jth ordered segment on the kth guide hair. With this notation, LHS uses a linear interpolation stencil to derive the displacement of rendered hair as:

$$\mathbf{x}_i - \mathbf{x}_i^0 = \sum_{k\in\mathcal{N}} \sum_j w_{ijk} \left(\mathbf{x}_{k,j} - \mathbf{x}_{k,j}^0\right), \tag{9}$$

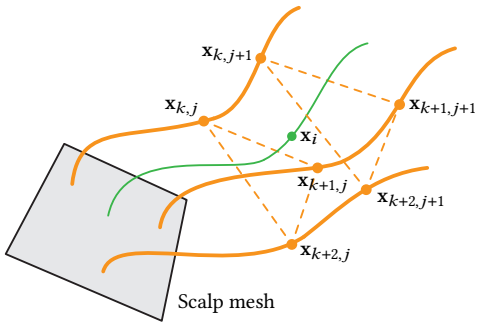

Fig. 3. *Conventional Linear Hair Skinning (LHS). Segments from three guide hairs (orange) form a triangular prism and are used to interpolate the rendered hair (green) in a linear skinning fashion.*

which is locally supported on the small neighborhood set $\mathcal{N}$ of adjacent guide hairs, and we again use superscript ^{0} to indicate rest pose. Fig. 3 demonstrates the conventional interpolation scheme using three guide hairs forming a triangular prism for linear skinning fashion. The barycentric weights w_{ijk} denote the contribution from the jth vertex of the kth guide hair to the ith vertex of a rendered hair, which is precomputed for each rendered hair vertex $\mathbf{x}_i^0$. Instead of interpolating positions, however, we adopt the linear interpolation stencil for stretching forces:

$$\mathbf{f}_i^{\text{sim}} = \sum_{k\in\mathcal{N}} \sum_j w_{ijk} \mathbf{f}_{k,j}^{\text{ss}}, \tag{10}$$

where $\mathbf{f}_{k,j}^{\text{ss}}$ is the force provided by the simulator on the jth vertex of the kth guide hair. The superscript $^{\text{sim}}$ denotes a variable provided by the simulator and used for reconstruction.

Our goal is then to find the set of segment positions and orientations that reproduce the target force $\mathbf{f}_i^{\text{sim}}$. This problem, however, is significantly under-determined in general since multiple sets of positions and orientations can reproduce the target force. To resolve this ambiguity, we utilize the observation that orientations typically converge very rapidly. As a result, non-zero net torques can contain spurious noises, especially in a less-accurate time-integration scheme such as XPBD [Macklin et al. 2016]. We thus decide to further constrain our problem by assuming that the net torque $\boldsymbol{\tau}_i^{\text{net}} = 0$ within each rendered hair strand. Note that individual bending and torque can still emerge as a part of our interpolation. We only specify the net torque at each segment. Put together, vertex positions can be computed as the solution that matches force with zero net torque under unit quaternion orientations. To write constraints more precisely, we have:

$$\forall i : \begin{cases} \mathbf{f}_i^{\text{ss}} \triangleq \frac{k^{\text{ss}}}{l_i}\left(\frac{\mathbf{x}_{i+1}-\mathbf{x}_i}{l_i} - \mathbf{d}_{i,3}\right) = \mathbf{f}_i^{\text{sim}} \\ \boldsymbol{\tau}_i^{\text{net}} \triangleq \boldsymbol{\tau}_i^{\text{ss}} + \boldsymbol{\tau}_{i,i-1}^{\text{bt}} + \boldsymbol{\tau}_{i,i}^{\text{bt}} = \lambda_i \mathbf{q}_i \\ \|\mathbf{q}_i\| = 1 \end{cases}, \tag{11}$$

where the Lagrangian multiplier λ_i is used to satisfy the unit quaternion constraints. As any extraneous torque along $\mathbf{q}_i$ cannot move the same unit quaternion $\mathbf{q}_i$ itself, the changing of λ_i does not affect the overall torque balance. Although these constraints are independent for each rendered hair, finding a solution for this nonlinear system is still highly non-trivial.

To devise an efficient solver, we first note that the first set of constraints, $\mathbf{f}_i^{\text{ss}} = \mathbf{f}_i^{\text{sim}}$, forms a tridiagonal linear system in terms of the vertices $\mathbf{x}_i$. Indeed, given the tangents $\mathbf{d}_{i,3}$, we can fix $\mathbf{x}_{i-1}$ and solve for $\mathbf{x}_i$ to satisfy the ith equation. With the first vertex $\mathbf{x}_0$ fixed on the scalp, the linear system is fully determined and can be solved starting at the hair root and working towards the hair tips, leading to the following succinct formula for the vertex positions:

$$\mathbf{x}_i = \mathbf{x}_0 + \sum_{m=0}^{i-1} l_m \left(\frac{\mathbf{f}_m^{\text{sim}} l_m}{k^{\text{ss}}} + \mathbf{d}_{m,3} \right). \tag{12}$$

The bulk of our solver then lies in finding the correct segment orientation with tangent $\mathbf{d}_{i,3}$, which satisfies our constraint $\boldsymbol{\tau}_i^{\text{net}} = \mathbf{0}$ and in turn fully determines $\mathbf{x}_i$.

Expanding the full equations for torque (Eq. 6, Eq. 7, and Eq. 8) and removing any scalar terms of $\mathbf{q}_i$, we arrive at the following equivalent nonlinear system of equations to Eq. 11:

$$\begin{cases} -2l_i \mathbf{f}_i^{\text{sim}} \mathbf{q}_i \mathbf{e}_3 + k^{\text{bt}} \phi_{i-1} \mathbf{q}_{i-1} \mathbf{q}_{i-1}^0 + k^{\text{bt}} \phi_i \mathbf{q}_{i+1} \overline{\mathbf{q}}_i^0 - \lambda_i \mathbf{q}_i = 0 \\ \|\mathbf{q}_i\| = 1 \end{cases} \tag{13}$$

in which $\mathbf{f}_i^{\text{sim}} \mathbf{q}_i \mathbf{e}_3$ is a 4D vector. Since we only use the solution for rendering hairs instead of simulation, we do not require solving Eq. 13 exactly. As such, we devise an approximate solution using local constraint projections in a Gauss-Seidel fashion. Instead of considering the entire global system at once, we satisfy the segment orientations and their associated constraints individually on a per-segment basis in each local step.

Although the global system is nonlinear, treating the segments locally allows us to take advantage of the linear nature of quaternion multiplication. We arrange the torque terms into a linear system of $\mathbf{q}_j$ and rewrite the torque-balancing condition as:

$$-2l_i \mathbf{f}_i^{\text{sim}} \mathbf{q}_i \mathbf{e}_3 \triangleq \mathbf{A}_i \mathbf{q}_i = \mathbf{b}_i \triangleq -k^{\text{bt}} \left(\phi_{i-1} \mathbf{q}_{i-1} \mathbf{q}_{i-1}^0 + \phi_i \mathbf{q}_{i+1} \overline{\mathbf{q}}_i^0 \right), \tag{14}$$

where the left-hand side is encoded as a 4×4 linear matrix $\mathbf{A}_i$, and the right-hand side is encoded as a 4D vector $\mathbf{b}_i$. If we assume that $\mathbf{q}_{i-1}$ and $\mathbf{q}_{i+1}$ are fixed, Eq. 13 takes the following form:

$$\begin{cases} (\mathbf{A}_i - \lambda_i \mathbf{I}_4)\, \mathbf{q}_i = \mathbf{b}_i \\ \|\mathbf{q}_i\| = 1. \end{cases} \tag{15}$$

Solving the 4-dimensional sub-system Eq. 15 yields an update rule for the orientation of a single rendered hair segment. Recall that solving this system exactly amounts to finding the scalar λ_i leading to a unit quaternion solution, which, in turn, requires solving the quartic polynomial:

$$\|(\mathbf{A}_i - \lambda_i \mathbf{I}_4)^{-1} \mathbf{b}_i\| = 1. \tag{16}$$

Although robust algorithms such as the bisection method exist for solving quartic polynomials, doing so for every segment of the rendered hair is highly inefficient on the GPU. As such, we instead propose a heuristic approach to first over-estimate λ_i by taking $\lambda_i = \|2l_i \mathbf{f}_i^{\text{sim}}\| + \|\mathbf{b}_i\|$, leading to a $\|\mathbf{q}_i\| < 1$, and then normalize $\mathbf{q}_i$. To explain this approach, we introduce the following theorem:

Theorem 4.1. *$\mathbf{A}_i$ is a symmetric matrix with two pairs of eigenvalues at $\pm\|2l_i \mathbf{f}_i^{sim}\|$. Furthermore, the inversion of $\mathbf{A}_i - \lambda_i \mathbf{I}_4$ takes the following closed form:*

$$\begin{cases} \boldsymbol{\nu} \triangleq -2l_i \mathbf{f}_i^{sim} \\ (\mathbf{A}_i - \lambda_i \mathbf{I}_4)^{-1} = \frac{1}{\|\boldsymbol{\nu}\|^2 - \lambda^2} \begin{pmatrix} \lambda + \boldsymbol{\nu}_2 & 0 & -\boldsymbol{\nu}_0 & \boldsymbol{\nu}_1 \\ 0 & \lambda + \boldsymbol{\nu}_2 & -\boldsymbol{\nu}_1 & -\boldsymbol{\nu}_0 \\ -\boldsymbol{\nu}_0 & -\boldsymbol{\nu}_1 & \lambda - \boldsymbol{\nu}_2 & 0 \\ \boldsymbol{\nu}_1 & -\boldsymbol{\nu}_0 & 0 & \lambda - \boldsymbol{\nu}_2 \end{pmatrix}. \end{cases}$$

Theorem 4.1 can be verified via the attached WxMaxima code. Using an Eigen analysis, we can see that setting $\lambda_i = \pm(\|2l_i \mathbf{f}_i^{\text{sim}}\| + \|\mathbf{b}_i\|)$ would always lead to a $\|\mathbf{q}_i\| < 1$, i.e., the magnitude of λ_i is over-estimated. Note that although the choice of either λ_i does not affect the torque-equilibrium condition, λ_i can affect the local stability of the hair strand near the configuration of $\mathbf{q}_i$. Indeed, if the interpolated result is used in a hypothetical simulation, a large positive λ_i would have a stabilizing effect that pushes a deviated hair strand back to the configuration $\mathbf{q}_i$, i.e., our approximate solution slightly exaggerates the stabilizing effect.

Put together, we propose the following closed-form solution for $\mathbf{q}_i$:

$$\begin{cases} \lambda_i \triangleq \|2l_i \mathbf{f}_i^{\text{sim}}\| + \|\mathbf{b}_i\| \\ \mathbf{q}_i \triangleq \frac{(\mathbf{A}_i - \lambda_i \mathbf{I}_4)^{-1} \mathbf{b}_i}{\|(\mathbf{A}_i - \lambda_i \mathbf{I}_4)^{-1} \mathbf{b}_i\|} \end{cases}, \tag{17}$$

which is very efficient to implement using the closed-form inversion of $\mathbf{A}_i - \lambda_i \mathbf{I}_4$ in Theorem 4.1. Although we slightly underestimate the effect of the target force using this λ_i, in practice, we find the impact to be minimal. Our approximate solver then only has to solve the sub-systems from root to tip until termination conditions hold. In the following, we propose several extensions to our interpolation framework to improve visual quality and performance.

4.2 Penetration Correction

Penetration correction can be necessary for protruding surface features, like ears and noses. Although our guide hair simulator already incorporates collisions, it cannot guarantee that interpolated hairs are penetration-free. One naive solution for resolving penetration is to project the final position outside any colliders using a Signed Distance Field (SDF), denoted as ψ. However, as shown in Fig. 13, projecting the final positions directly often invariably causes similar visual artifacts in rendered hair geometry, leading to poor render quality. This is because the orientations of hair segments are not properly modified to reflect the shift in position due to the projection.

To make collisions handling for individual rendered hair to be properly reflected in hair reconstruction, we introduce a SDF-based penetration energy E_i^{pen} with stiffness k^{pen}, which is defined vertex-wise as:

$$E_i^{\text{pen}} = \frac{k^{\text{pen}}}{2} \min(0, \psi(\mathbf{x}_i))^2. \tag{18}$$

We then plug the position reconstruction Eq. 12 into E_i^{pen}, casting it as a function on the segment orientations. As a result, the vertex-wise penetration energy imposes torque locally as:

$$\boldsymbol{\tau}_{i,i}^{\text{pen}} = -2k^{\text{pen}} l_i \min(0, \psi(\mathbf{x}_i)) \nabla \psi(\mathbf{x}_i) \mathbf{q}_i \mathbf{e}_3. \tag{19}$$

Fig. 4. *Given guide hairs, while LHS introduces zig-zag artifacts, force-based reconstruction would lead to undesired hair shapes that are far from the guide hairs. At the rightmost, we show that simply blending those two solutions in half would not resolve the artifacts.*

Following the same logic as Sec. 4.1, we can then incorporate this term into $\boldsymbol{\tau}_i^{\text{ss}}$ in Eq. 11 by the following redefinition:

$$\boldsymbol{\tau}_i^{\text{net}} \triangleq \boldsymbol{\tau}_i^{\text{ss}} + \boldsymbol{\tau}_{i,i-1}^{\text{bt}} + \boldsymbol{\tau}_{i,i}^{\text{bt}} + \boldsymbol{\tau}_{i,i}^{\text{pen}} = \lambda_i \mathbf{q}_i. \tag{20}$$

As we typically only use a single Gauss-Seidel sweep in practice, the SDF value of each vertex can be unknown before positions are computed. Therefore, we choose to first solve each vertex assuming that all $\psi(\mathbf{x}_j) = 0$, i.e., without considering E^{pen}. We then query $\psi(\mathbf{x}_j)$ at the reconstructed positions according to Eq. 12, before performing a second solve with E^{pen}, if needed. In practice, this procedure can be implemented very efficiently on GPU with marginal overhead. While this approach can inherit known temporal coherence issues from sharp SDF discontinuities, we found the performance gain advantageous for our particular use case. In general, we found the issues with SDF-caused temporal incoherence to be minor to negligible in all our examples, as shown in our supplemental materials.

4.3 Drift Correction

Due to the discretized nature of our hair and the approximate nature of our solutions, errors will accumulate in each hair segment as we solve from root to tip. Since internal force is a local measure, it is translation invariant and does not penalize translations. As such, some hairs can drift undesirably far from the guide hairs along the strand. This is especially problematic for strands with little internal stress as the entire strand becomes dependent on the initial root orientation. While we want to preserve the shape of the hair, we also want the interpolated hair shapes to be consistent with whose of the guide hairs. A common approach would be to correct this error by directly blending a stable solution with the output. Unfortunately, directly blending the reconstructed position of our method with that from LHS would re-introduce the artifacts of conventional interpolation schemes and diminish the benefit of our method (see Fig. 4).

Instead, we propose a blending scheme within our force-reconstruction framework. Note that if the interpolated force $\mathbf{f}_i^{\text{sim}} = 0$, Eq. 15 has the following trivial solution, denoted as $\hat{\mathbf{q}}_i$:

$$\hat{\mathbf{q}}_i \triangleq \frac{-\mathbf{b}_i}{\|\mathbf{b}_i\|} \quad \text{and} \quad \hat{\mathbf{d}}_{i,3} \triangleq \hat{\mathbf{q}}_i \mathbf{e}_3 \bar{\hat{\mathbf{q}}}_i. \tag{21}$$

Plugging this solution into the definition of $\mathbf{f}_i^{\text{ss}}$ in Eq. 11, we derive an estimation of the stretch force $\hat{\mathbf{f}}_i^{\text{ss}}$ as:

$$\hat{\mathbf{f}}_i^{\text{ss}} \triangleq \frac{k^{\text{ss}}}{l_i}\left(\frac{\mathbf{x}'_{i+1} - \mathbf{x}_j}{l_i} - \hat{\mathbf{d}}_{i,3}\right), \tag{22}$$

where $\mathbf{x}'_{i+1}$ can be any desirable position. For example, $\mathbf{x}'_{i+1}$ can be taken from LHS-based prior works, e.g., Chai et al. [2017], to address finer collisions. An interesting consequence is that our force-based reconstruction can then act as a filter for physically plausible hair geometries for any given $\mathbf{x}'_{i+1}$. However, here, we choose to use a fast static weight LHS scheme with weights based on the closest polyline segments. This is to avoid the requirement for training data and for performance. Our drift correction is then achieved by blending $\hat{\mathbf{f}}_i^{\text{ss}}$ and $\mathbf{f}_i^{\text{sim}}$ as:

$$\mathbf{f}_i^{\text{sim}} \leftarrow \alpha \hat{\mathbf{f}}_i^{\text{ss}} + (1-\alpha)\mathbf{f}_i^{\text{sim}}, \tag{23}$$

where α is taken as a small blending weight (of $0.05 \sim 0.1$ in our examples).

4.4 Warm Starting

As we use a very small number of iterations to solve Eq. 13 in practice for performance, the quality of the resulting quaternion significantly relies on a reasonable initialization. One typical solution is to use the orientation from a previous frame as the initial guess. However, significant overheads would have to be introduced to write back and update new solutions. Instead, we would like to opt for a solution that allows us to discard updates without performing memory writes. Our strategy is to assume that the bending energy of the next segment lies in its local minimum, which is reached when:

$$\mathbf{q}_{i+1}\bar{\mathbf{q}}_i^0 = \mathbf{q}_i. \tag{24}$$

Since the torque along $\mathbf{q}_i$ does not affect the imaginary components, this simply results in the corresponding term in Eq. 13 to be taken out. This approach results in a faster hair interpolation scheme.

Finally, as an interesting side effect of our choice of warm starting, we can safely assume that $\phi = 1$ at all times in $\mathbf{C}_j^{\text{bt}}$. When using our choice of λ, $\mathbf{A}_i - \lambda_i \mathbf{I}_4$ will always have negative eigenvalues that result in a solution closer to the positive pole, making our implementation slightly less complex in practice.

5 RESULTS

We implement both our simulation and interpolation using C++ and CUDA. Unless otherwise stated, we measure performance on an NVIDIA RTX 4090 GPU. For our simulation, we measure about 0.2 ms per 1 ms timestep with only 128 guide hairs. For our interpolation, we split the algorithm into two CUDA kernel calls. The first kernel call computes the internal force on each guide hair segment. The second kernel call then uses one thread per hair to compute their final positions in a single pass from the root to the tip. For drift correction and as a baseline, we implement a traditional static weight LHS scheme with weights based on the closest polyline segments.

Algorithm 1: Pseudocode for our hair interpolation looped over one hair strand.

foreach $i \in [0, \textit{Num Vertex in Hair} - 1)$ **do**

// Compute target force

$\mathbf{f}_i^{\text{sim}} \leftarrow$ LHS based force (Eq. 10)

// Compute drift correction

$\mathbf{x}'_{i+1} \leftarrow$ LHS based position (Eq. 9)

$\hat{\mathbf{f}}_i^{\text{ss}} \leftarrow$ Eq. 22 given $\mathbf{x}'_{i+1}$

$\mathbf{f}_i^{\text{sim}} \leftarrow (1-\alpha)\mathbf{f}_i^{\text{sim}} + \alpha\hat{\mathbf{f}}_i^{\text{ss}}$

// Solve for λ

$\mathbf{b}_i \leftarrow -k^{\text{bt}}\phi_{i-1}\mathbf{q}_{i-1}\mathbf{q}_{i-1}^0$

$\mathbf{v} \leftarrow -2l_i\mathbf{f}_i^{\text{sim}}$

$\lambda \leftarrow \|\mathbf{v}\| + \|\mathbf{b}_i\|$

// Use Theorem 4.1 given $\mathbf{v}$, $\mathbf{b}_i$, and normalize

$\mathbf{q}_i \leftarrow [\mathbf{A}_i - \lambda_i\mathbf{I}_4]^{-1}\mathbf{b}_i$

$\mathbf{q}_i \leftarrow \mathbf{q}_i/\|\mathbf{q}_i\|$

// Solve for $\mathbf{x}_{i+1}$

$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + l_i\left[\mathbf{f}_i^{\text{sim}}l_i/k^{\text{ss}} + \mathbf{d}_{i,3}\right]$

// Check SDF for collision

if $\psi(\mathbf{x}_{i+1}) < 0$ **then**

// Update $\mathbf{v}$ and resolve

$\mathbf{v} \leftarrow \mathbf{v} - 2k^{\text{pen}}l_i\psi(\mathbf{x}_{i+1})\nabla\psi(\mathbf{x}_{i+1})$

$\lambda \leftarrow \|\mathbf{v}\| + \|\mathbf{b}_i\|$

$\mathbf{q}_i \leftarrow [\mathbf{A}_i - \lambda_i\mathbf{I}_4]^{-1}\mathbf{b}_i/\|[\mathbf{A}_i - \lambda_i\mathbf{I}_4]^{-1}\mathbf{b}_i\|$

$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + l_i\left[\mathbf{f}_i^{\text{sim}}l_i/k^{\text{ss}} + \mathbf{d}_{i,3}\right]$

end

end

In contrast to prior works that use training [Chai et al. 2017; Lyu et al. 2022], our method requires no pre-computation, no training data, and is simple to implement. In particular, the force reconstruction only represents some dozen extra lines of code on top of any existing interpolation, as shown in Algorithm 1. Correspondingly, the overhead is extremely small. To further maximize performance, we packed the data required for force reconstruction together. For each vertex, this consists of $\mathbf{q}_{r,j}$, l_j, and k^{norm}, which is the normalized stiffness computed by $k^{\text{norm}} = k^{\text{bt}}/k^{\text{ss}}$. Conceptually, we divide both sides of Eq. 13 by k^{ss} such that we only need to load one stiffness value for each segment. For all our scenes, k^{pen} is set to $1e4$ as if it were a simple quadratic contact penalty energy.

5.1 Comparisons to LHS

We demonstrate our method on various real-world hairstyles captured from Unreal Metahumans [Epic 2024], multi-view 3D reconstruction [Luo et al. 2013], and computed tomography [Shen et al. 2023]. As our interpolation is designed specifically for real-time applications, we use low-resolution simulations of only 128 to 256 guide hairs each. This proves to be a very challenging use case for traditional LHS, especially in cases involving curly hair.

Fig. 1 demonstrates this challenge. As the curls stretch, bend, and unwind, the non-linear motion causes significant artifacts in the fine geometry of the hair. In contrast, our method can efficiently recreate plausible interpolated rendered hairs that visually preserve the original hairstyle. It is true even when the same hairstyle is simulated with 16× the number of guide hairs, as shown in Fig. 5, which takes 1.6 ms per frame for the simulation and is too expensive for most real-time applications. Since the issue fundamentally arises from the shape of each individual rendered hair, adding more guide hairs does little to rectify the artifacts. On the other hand, our method can produce artifact-free interpolations with extremely sparse guide hairs. Note that, as our interpolation solves for the quasistatic shape of rendered hairs with interpolated forces, it does not depend on or affect the dynamic state of the simulator.

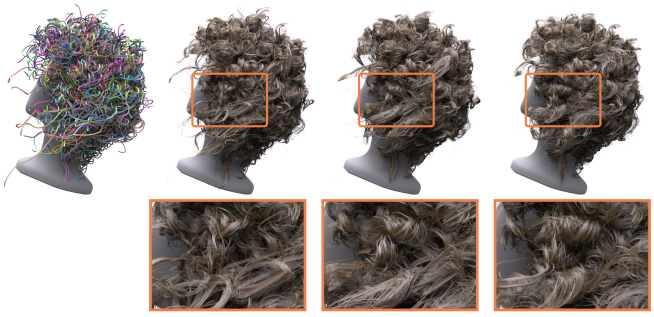

Fig. 5. *Artifacts can still easily occur when the hairstyle is simulated with a large number of guide hairs. Here, we use 2048 guide hairs.*

Fig. 6 demonstrates the robustness of our method to hair length. In this example, traditional LHS creates significant artifacts as the long strands unwind. Even only using a single guide hair for each rendered hair (LHS single), the simple motion of unwinding distorts the shapes of the interpolated strands into unrecognizable zig-zags. Our method produces artifact-free interpolations while remaining stable despite the fast and complex swinging motions of long hairs.

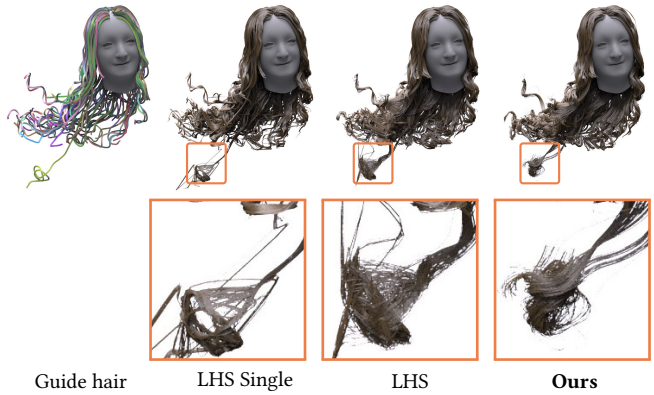

Fig. 6. *When long curly hair unwinds and rotates in unpredictable ways, even interpolating rendered hair with only one guide hair can cause significant artifacts in hair geometry. Our method is able to handle this case robustly and efficiently.*

The visual improvements of our method are also not limited to curly hair. The straight hairstyle in Fig. 7 exhibits significant

artifacts with LHS, as guide hairs separate. Although this can be remedied by restricting each rendered hair to only use a single guide hair (as in Fig. 7 LHS single), the result creates obvious clumping in motion. In cases where this clumping is undesirable, obtaining robust weights without relying on training data is highly difficult. Despite relying on the same exact interpolation method in both force interpolation and drift correction, our method successfully avoids the many artifacts from LHS in all our examples.

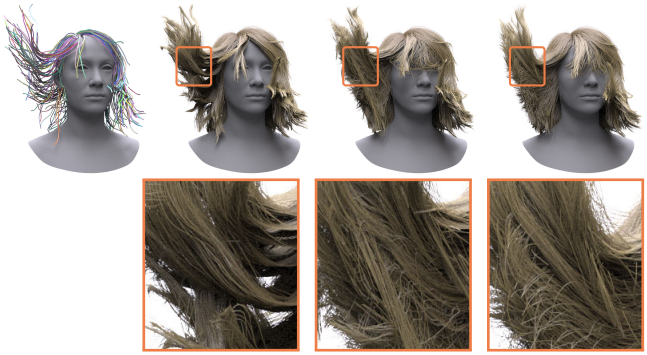

Fig. 7. *Even with straight hairstyles, both one (LHS single) and triple (LHS) guides per rendered hair introduce odd clumping. Despite the reliance on the same LHS scheme for interpolating forces, our method can effectively remove the various artifacts in geometry.*

To further demonstrate the robustness of our method under realistic motions, we test our interpolation using motion capture and animation data in Fig. 8 and Fig. 10. As shown, animations with large and sudden head motions very rapidly create chaotic movements that can be unaccounted for during training or pre-computation. Despite this, our interpolation can produce realistic interpolations with very few guide hairs required. Fig. 9 demonstrates that visual artifacts caused by LHS still exist even with a slow motion, while our method remains resistant to geometric artifacts.

Performance. We present the cost of our interpolation in Table 1. Compared to the LHS interpolation that our examples are based on, our method only presents a 20% overhead in total interpolation time. Even for the largest example with over 100K hair strands, our method never required more than half a millisecond, satisfying the real-time requirement.

Table 1. *Performance of our method against LHS using three guide hairs. Timings in microseconds (μs) per frame are measured on an Nvidia RTX 4090 GPU. Overhead is measured by our method compared against LHS using the same number of guide hairs per rendered hair. Guide and rendered hairs have the same number of vertices per strand.*

	Guides/Hairs	Vertices	LHS	Ours	Overhead
Josh (Fig. 1)	128 / 106,686	1,706,976	283 μs	346 μs	22%
Alice (Fig. 6)	128 / 30,307	1,939,648	360 μs	425 μs	17%
Hadley (Fig. 7)	256 / 64,663	1,034,608	163 μs	199 μs	22%
Danielle (Fig. 8)	256 / 46,706	747,296	111 μs	137 μs	23%

Ablation studies. As demonstrated in Fig. 11, the decision not to use the solution of prior frames does not significantly impact the quality of the final results. The overall appearance of the hairstyle remains faithful to the original from the first iteration and onwards. Even fully allowing our method to converge, the differences are minor. Furthermore, by eliminating the extra data movement, we observe an over 52% speedup in interpolation time. Compared to traditional LHS, this represents an approximate drop in overhead from 85% to our 20%.

We perform further ablation studies in Fig. 12 and 13. In Fig. 12, without our drift correction, strands can easily veer off course, depending on the initial orientations at the hair root. This is especially true with hairs experiencing weightlessness. With drift correction, this issue is largely resolved with minimal degradation in final hair geometry.

Fig. 13 demonstrates the importance of incorporating our penetration energy. For sharp SDF discontinuities, simple position projections can fail to improve the result meaningfully compared to no collisions. In contrast, our penetration energy can consider the rest shapes for a higher-quality result.

5.2 Comparison with Neural Hair Interpolation

We compare our interpolation method with the only available open-source neural interpolation work [Shen et al. 2023], in which a variational autoencoder (VAE) is pre-trained to encode guide hairs into latent vectors. These latent vectors are then interpolated based on barycentric coordinates on the scalp and decoded into rendered hairs. Fig. 14 presents a side-by-side comparison of a curly hairstyle interpolated using 2048 guide hairs. The results show that our interpolation can faithfully reproduce the hairstyle, while the neural interpolation simply enriches the hair count through interpolation without considering the rest pose of the hairstyle. Additionally, as an offline method, Shen et al. [2023] take 358 seconds on an Nvidia RTX 3080 to interpolate 59220 hair strands with 256 vertices per strand, several orders of magnitude slower than ours. Furthermore, when compared to the real-time neural interpolation proposed by Lyu et al. [2022], which takes 151 ms to interpolate 1M vertices, our method requires only 1.6 ms to interpolate 1M vertices (for Hadley) when measured on the same hardware (Nvidia GTX 1080), achieving about two orders of magnitude faster results. Notably, our method does not necessitate the pre-simulation of a large dataset for training.

6 CONCLUSION

We have presented a novel, robust, and efficient physical-guided hair interpolation scheme for real-time applications. Our method uses LHS to interpolate the physical parameters and forces already readily available in real-time simulations, which are then used to reconstruct the hair vertex positions. This results in an interpolator that requires no training data and no pre-computation. We have demonstrated the effectiveness of our approach in challenging real-world scenarios with interpolation times far faster than alternatives, affirming our suitability for real-time applications.

Future work and limitations. An interesting future direction would be generalizing our method to other hair simulation frameworks, e.g., discrete elastic rods [Bergou et al. 2008] and

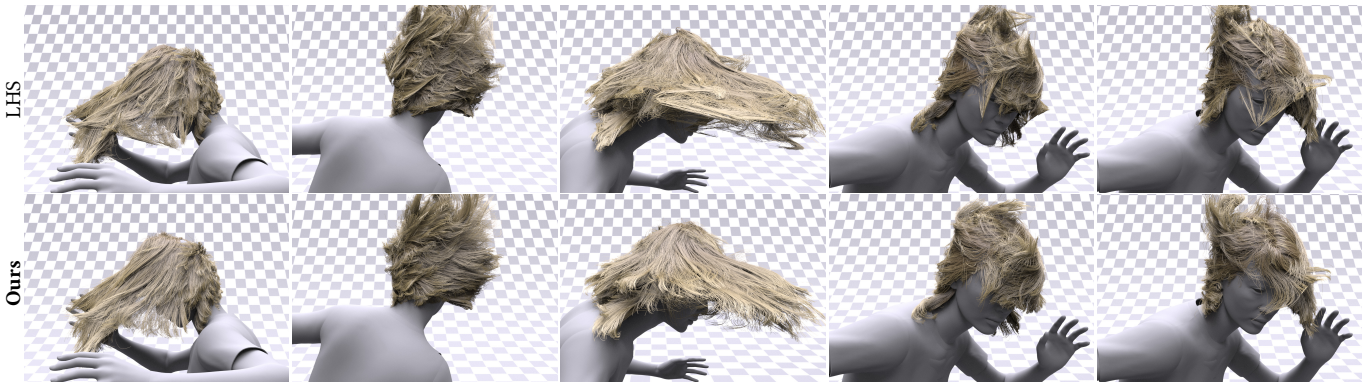

Fig. 8. *Large separations in guide hairs can often cause errors in LHS weights to amplify. Although we do not use any dynamic guide or weight selection, our method remains resistant to geometric artifacts.*

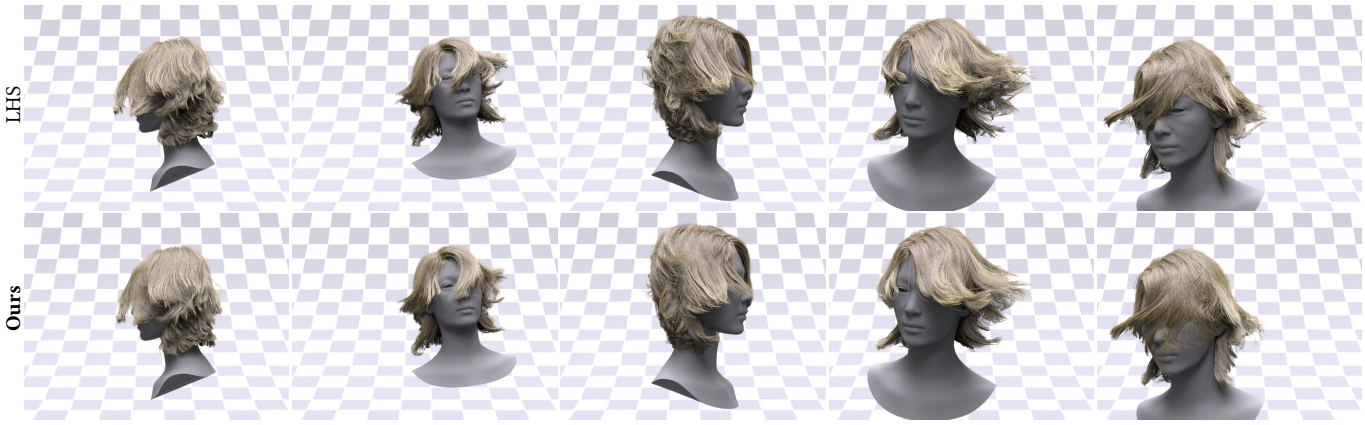

Fig. 9. *Even with a slow motion, visual artifacts caused by LHS still exist, while our method remains resistant to geometric artifacts robustly.*

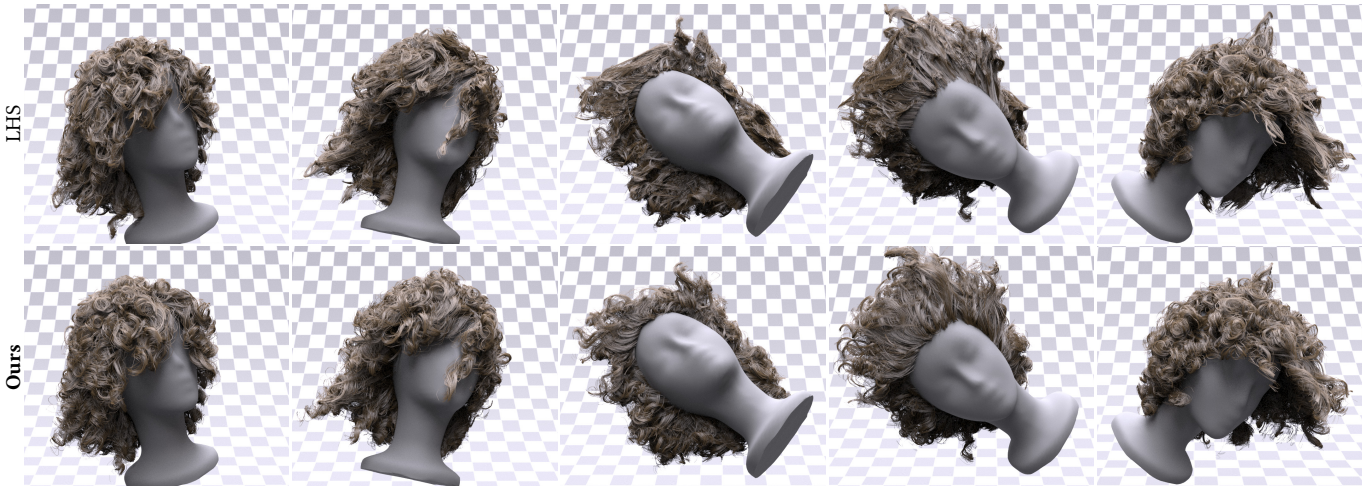

Fig. 10. *Despite only using* 128 *guide hairs, our interpolation remains robust under the large perturbations common in motion capture data.*

altitude springs [Selle et al. 2008]. Although our core ideas still apply, significant work is needed to recreate our efficient constraint solver for each framework, which is non-trivial. Our interpolation can also be incompatible with sag-free initialization methods [Hsu et al. 2023] that modify the rest shapes of hair strands to eliminate sagging at the beginning of simulations. Given that our method

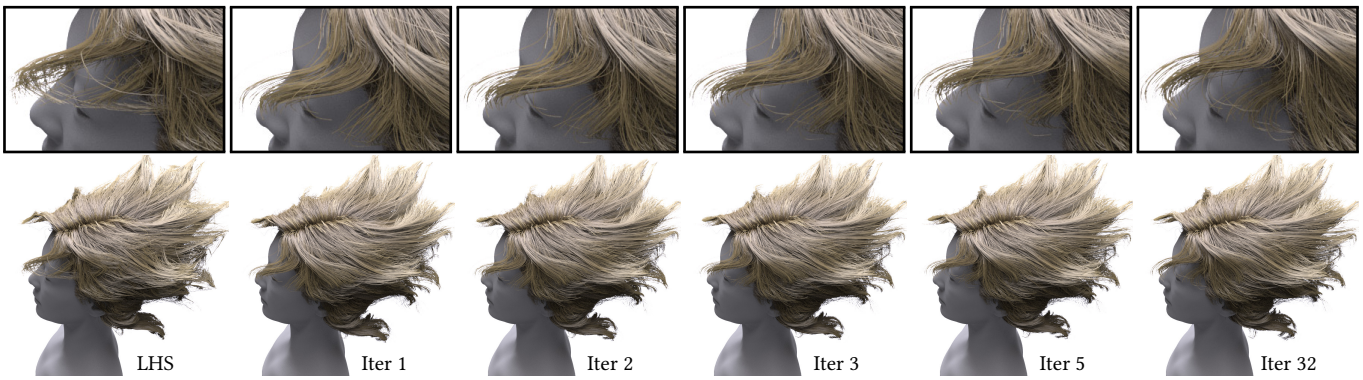

Fig. 11. *We show the result of our interpolation at various iterations starting with our initial guess in Sec. 4.4. The differences are minor, although there are some minor differences between the first and final converged iteration. Most importantly, compared to the leftmost LHS result, our solutions are artifact-free from the first iteration onwards, demonstrating the effectiveness of our initial guess.*

Fig. 12. *Comparison between LHS and ours with different drift correction blending parameters. Without drift correction, the flow of hairs that experience weightlessness (with little internal stress) become almost completely dependent on the root orientation.*

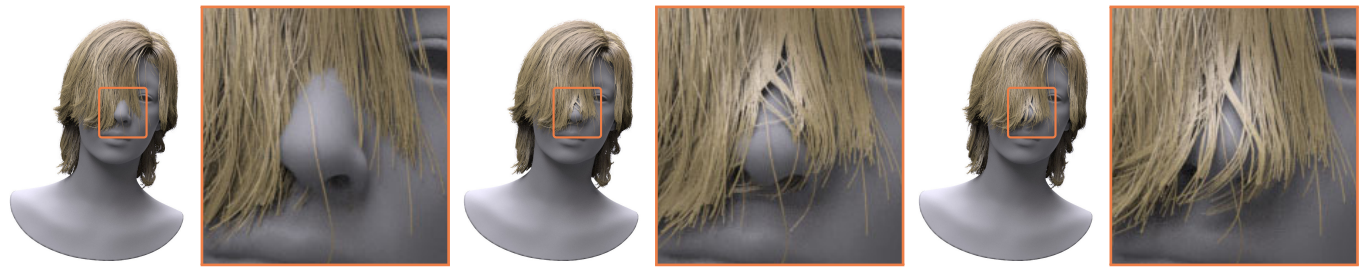

Fig. 13. *Directly projecting positions may result in vertices on alternating sides of a protrusion. By taking bending energies into consideration, our SDF penalty energy avoids this issue and produces strands that favor a consistent projection direction.*

Fig. 14. *Comparison with neural interpolation [Shen et al. 2023]: given 2048 guide hairs, our interpolation can faithfully reproduce the hairstyle, while the neural interpolation simply enriches the hair count through interpolation without considering the rest pose of the hairstyle.*

relies on the natural rest shape of the hair, any modifications introduced by sag-free initialization techniques would require extra consideration during initialization to preserve the style and ensure robustness. Finally, our method can exhibit less natural solutions due to the approximate solution scheme. For real-time performance, in particular, we do not consider the inter-segment coupling due to the penetration constraints. Efficient and more accurate solution techniques are left as future works.

ACKNOWLEDGMENTS

We would like to thank our colleagues from LightSpeed Studios, Fengquan Wang and Dong Li, for their project support. This project was supported in part by NSF grant #1956085.

REFERENCES

Ken-ichi Anjyo, Yoshiaki Usami, and Tsuneya Kurihara. 1992. A Simple Method for Extracting the Natural Beauty of Hair. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*. Association for Computing Machinery, New York, NY, USA, 111–120.

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable object animation using reduced optimal control. *ACM Trans. Graph.* 28, 3, Article 53 (jul 2009), 9 pages.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete Elastic Rods. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 63, 12 pages.

Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévêque. 2006. Super-Helices for Predicting the Dynamics of Natural Hair. *ACM Trans. Graph.* 25, 3 (jul 2006), 1180–1187.

Florence Bertails, Sunil Hadap, Marie-Paule Cani, Ming Lin, Tae-Yong Kim, Steve Marschner, Kelly Ward, and Zoran Kačić-Alesić. 2008. Realistic Hair Simulation: Animation and Rendering. In *ACM SIGGRAPH 2008 Classes* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 89, 154 pages.

Menglei Chai, Changxi Zheng, and Kun Zhou. 2014. A Reduced Model for Interactive Hairs. *ACM Trans. Graph.* 33, 4, Article 124 (jul 2014), 11 pages.

Menglei Chai, Changxi Zheng, and Kun Zhou. 2017. Adaptive Skinning for Interactive Hair-Solid Simulation. *IEEE Transactions on Visualization and Computer Graphics* 23, 7 (July 2017), 1725–1738.

Johnny T. Chang, Jingyi Jin, and Yizhou Yu. 2002. A Practical Model for Hair Mutual Interactions. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 73–80.

Gilles Daviet. 2020. Simple and Scalable Frictional Contacts for Thin Nodal Objects. *ACM Trans. Graph.* 39, 4, Article 61 (aug 2020), 16 pages.

Gilles Daviet. 2023. Interactive Hair Simulation on the GPU Using ADMM. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 24, 11 pages.

Epic. 2024. Unreal Engine. https://www.unrealengine.com

Yun (Raymond) Fei, Qi Guo, Rundong Wu, Li Huang, and Ming Gao. 2021. Revisiting Integration in the Material Point Method: A Scheme for Easier Separation and Less Dissipation. *ACM Trans. Graph.* 40, 4, Article 109 (jul 2021), 16 pages.

Stefan Fröhlich and Mario Botsch. 2011. Example-Driven Deformations Based on Discrete Shells. *Computer Graphics Forum* 30, 8 (2011), 2246–2257.

Yang Guang and Huang Zhiyong. 2002. A method of human short hair modeling and real time animation. In *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.* IEEE, New York, NY, USA, 435–438.

Xuchen Han, Theodore F. Gast, Qi Guo, Stephanie Wang, Chenfanfu Jiang, and Joseph Teran. 2019. A Hybrid Material Point Method for Frictional Contact with Diverse Materials. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 17 (July 2019), 24 pages.

B. Heeren, M. Rumpf, P. Schröder, M. Wardetzky, and B. Wirth. 2014. Exploring the geometry of the space of shells. In *Proceedings of the Symposium on Geometry Processing* (Cardiff, United Kingdom) *(SGP '14)*. Eurographics Association, Goslar, DEU, 247–256.

Jerry Hsu, Tongtong Wang, Zherong Pan, Xifeng Gao, Cem Yuksel, and Kui Wu. 2023. Sag-Free Initialization for Strand-Based Hybrid Hair Simulation. *ACM Trans. Graph.* 42, 4, Article 74 (jul 2023), 14 pages.

Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2006. Subspace Gradient Domain Mesh Deformation. *ACM Trans. Graph.* 25, 3 (jul 2006), 1126–1134.

Jin Huang, Yiying Tong, Kun Zhou, Hujun Bao, and Mathieu Desbrun. 2010. Interactive shape interpolation through controllable dynamic deformation. *IEEE Transactions on Visualization and Computer Graphics* 17, 7 (2010), 983–992.

Li Huang, Fan Yang, Chendi Wei, Yu Ju (Edwin) Chen, Chun Yuan, and Ming Gao. 2023. Towards Realtime: A Hybrid Physics-Based Method for Hair Animation on GPU. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 43 (aug 2023), 18 pages.

Chuan Koon Koh and Zhiyong Huang. 2001. A Simple Physics Model to Animate Human Hair Modeled in 2D Strips in Real Time. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation* (Manchester, UK). Springer-Verlag, Berlin, Heidelberg, 127–138.

T. Kugelstadt and E. Schömer. 2016. Position and Orientation Based Cosserat Rods. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Zurich, Switzerland) *(SCA '16)*. Eurographics Association, Goslar, DEU, 169–178.

Minjae Lee, David Hyde, Michael Bao, and Ronald Fedkiw. 2019. A Skinned Tetrahedral Mesh for Hair Animation and Hair-Water Interaction. *IEEE Transactions on Visualization and Computer Graphics* 25, 3 (2019), 1449–1459.

Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. 2004. Seamless Image Stitching in the Gradient Domain. In *Computer Vision - ECCV 2004*, Tomás Pajdla and Jiří Matas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 377–389.

Linjie Luo, Hao Li, and Szymon Rusinkiewicz. 2013. Structure-aware hair capture. *ACM Trans. Graph.* 32, 4, Article 76 (jul 2013), 12 pages.

Qing Lyu, Menglei Chai, Xiang Chen, and Kun Zhou. 2022. Real-Time Hair Simulation With Neural Interpolation. *IEEE Transactions on Visualization and Computer Graphics* 28, 4 (April 2022), 1894–1905.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. ACM, New York, NY, USA, 49–54.

Aleka McAdams, Andrew Selle, Kelly Ward, Eftychios Sifakis, and Joseph Teran. 2009. Detail Preserving Continuum Simulation of Straight Hair. *ACM Trans. Graph.* 28, 3, Article 62 (jul 2009), 6 pages.

Dominik L. Michels, J. Paul T. Mueller, and Gerrit A. Sobottka. 2015. A Physically Based Approach to the Accurate Simulation of Stiff Fibers and Stiff Fiber Meshes. *Comput. Graph.* 53, PB (dec 2015), 136–146.

Robert E. Rosenblum, Wayne E. Carlson, and Edwin Tripp III. 1991. Simulating the structure and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation* 2, 4 (1991), 141–148.

Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. 2022. Neural Strands: Learning Hair Geometry And Appearance From Multi-View Images. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII* (Tel Aviv, Israel). Springer-Verlag, Berlin, Heidelberg, 73–89.

Christian Schumacher, Bernhard Thomaszewski, Stelian Coros, Sebastian Martin, Robert Sumner, and Markus Gross. 2012. Efficient simulation of example-based materials. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Goslar, DEU, 1–8.

Andrew Selle, Michael Lentine, and Ronald Fedkiw. 2008. A Mass Spring Model for Hair Simulation. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–11.

Yuefan Shen, Shunsuke Saito, Ziyan Wang, Olivier Maury, Chenglei Wu, Jessica Hodgins, Youyi Zheng, and Giljoo Nam. 2023. CT2Hair: High-Fidelity 3D Hair Modeling Using Computed Tomography. *ACM Trans. Graph.* 42, 4, Article 75 (jul 2023), 13 pages.

Vanessa Sklyarova, Jenya Chelishev, Andreea Dogaru, Igor Medvedev, Victor Lempitsky, and Egor Zakharov. 2023. Neural Haircut: Prior-Guided Strand-Based Hair Reconstruction. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*. IEEE, New York, NY, USA, 1–11.

Arunachalam Somasundaram. 2015. Dynamically Controlling Hair Interpolation. In *ACM SIGGRAPH 2015 Talks* (Los Angeles, California) *(SIGGRAPH '15)*. Association for Computing Machinery, New York, NY, USA, Article 36, 1 pages.

J. Spillmann and M. Teschner. 2007. CoRdE: Cosserat Rod Elements for the Dynamic Simulation of One-Dimensional Elastic Objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 63–72.

Nobuyuki Umetani, Ryan Schmidt, and Jos Stam. 2015. Position-Based Elastic Rods. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 21–30.

P. Volino and N. Magnenat-Thalmann. 2006. Real-time animation of complex hairstyles. *IEEE Transactions on Visualization and Computer Graphics* 12, 2 (2006), 131–142.

T. Winkler, J. Drieseberg, M. Alexa, and K. Hormann. 2010. Multi-Scale Geometry Interpolation. *Computer Graphics Forum* 29, 2 (2010), 309–318.

Kui Wu and Cem Yuksel. 2016. Real-Time Hair Mesh Simulation. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Redmond, Washington) *(I3D '16)*. Association for Computing Machinery, New York, NY, USA, 59–64.

Cem Yuksel, Scott Schaefer, and John Keyser. 2009. Hair Meshes. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–7.

Cem Yuksel and Sarah Tariq. 2010. Advanced Techniques in Real-Time Hair Rendering and Simulation. In *ACM SIGGRAPH 2010 Courses* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 1, 168 pages.

Changxi Zheng. 2013. One-to-Many: Example-Based Mesh Animation Synthesis. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Anaheim, California) *(SCA '13)*. Association for Computing Machinery, New York, NY, USA, 145–153.