https://doi.org/10.1016/j.gmod.2020.101077

Real-Time Hair Simulation with Heptadiagonal Decomposition on Mass Spring System

Jianwei Jiang[a], Bin Sheng[a,*], Ping Li[b], Lizhuang Ma[a], Xin Tong[c], Enhua Wu[d,e,*]

[a]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China*
[b]*Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China*
[c]*Microsoft Research Asia, Beijing, China*
[d]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China*
[e]*Faculty of Science and Technology, University of Macau, Macau, China*

Abstract

Simulating detailed dynamic hairs in real time is a challenging problem. Existing methods either simplify the strand dynamics or reduce the degrees of freedom at the cost of rich motion details. We present a real-time simulation for animating hair with high fidelity details. Our approach efficiently captures the inextensibility, bending and torsion strand mechanics, while presenting the stiction/repulsion and detailed real-time collision effects. To efficiently capture self-interactions, we factorize the phenomenon into a coarse, globally coupled volumetric, and detailed collision view. The coarse behaviors are solved with an Eulerian method via position-based density control, while detailed collisions are efficiently handled with temporal coherent link updates. We further provide a fast implicit integration via heptadiagonal matrix decomposition, which provides two to three orders of magnitude of acceleration to traditional methods. The efficiency and effectiveness of our method is validated by simulating variant motions of hair in various styles.

Keywords: Hair, real-time simulation, mass spring, time integration, GPU parallel computing.

1. Introduction

Human hair exhibits complicated dynamic behaviors and rich motion details as it moves with head/wind or interacts with other objects such as a human body. Faithfully simulating the detailed hair dynamics in real time is important for conveying realistic human characters in games and virtual reality. However, it is still a challenging task due to the large number of hair strands and self interactions. Several methods, such as [1, 2] and [3, 4, 5] have been proposed for generating high-fidelity hair animations. However, they all require expensive computations and are not applicable to real-time usage. Recent studies have focused on interactively simulating full hair geometry. These methods attempt to use a simplified mechanics scheme (e.g., [6, 7, 8]) or reduce the DoFs (degrees of freedoms) for hair simulation (e.g., [9, 10, 11]). These models either lack correct physical hair dynamics or suffer from other visual artifacts. For example, the constraint-based methods proposed in [6, 7] cannot correctly simulate curly hair, therein having to keep the curliness as a hard constraint along the centerline. Interpolation schemes such as [10] cannot handle complicated fuzzy hair geometries, and visual artifacts can be found by

[*]Corresponding authors
Email addresses: shengbin@sjtu.edu.cn (Bin Sheng), ehwu@um.edu.mo (Enhua Wu)

combining them with self-interactions. For instance, two similar interpolated hairs clumped with different wisps of hair strands.

In this paper, we present a real-time detail-preserving method for simulating high fidelity hair on a full scale. Our method is capable of capturing the inextensibility, curliness, bending and torsion effects of strand dynamics under a limited time budget. For self interactions, the coarse volumetric preservation is performed by adopting PBD(position-based dynamics) density control method. Details are further enhanced with hair-hair collisions. Finally, we adopt a two-stage velocity and position update strategy to correctly model static friction in the hair-solid collision stage. For strand dynamics integration, we utilize the hair representation model proposed in [12], where stretching, bending and torsion effects are addressed with a set of connected springs. Our method first reduces the implicit Euler into a linear problem and optimizes it by taking advantage of the local connectivity of the strand model, allowing us to exactly solve the problem with only a forward and backward sweep based on decomposition on the heptadiagonal matrix. Our work makes the following main contributions as:

- A real-time framework that produces high-fidelity hair dynamics, volumetric phenomena, and detailed self-collisions. The simulation results are comparable to the state-of-the-art interactive methods, where single hair strand behaviors either lack realism, or additional interpolation is needed to satisfy the time budget constraint.
- An interactive hybrid Eulerian/Lagrangian for handling self-interactions with detailed preservation. The coarse volumetric preservation is handled by a PBD density control strategy and the detailed hair collisions are further resolved with temporal coherent link update. Such a scheme allows us to provide high-resolution collision detection and response with efficient responses.
- A time-optimized implicit integration method solved via linearization on Newton method and heptadiagonal matrix decomposition. The computation is efficient and highly-scalable. We achieved two to three orders of magnitude speedup compared to the original global PCG integration via the GPU parallelism.

Our method greatly reduces the computation cost and can be efficiently implemented on GPU with good scalability. We evaluate the efficiency and effectiveness of our method for animating hairs with various types of strands and overall styles. The results show that our method can reproduce large-scale hair motions with details in real time.

2. Related Work

Hair simulation has been well studied for decades since [13, 14]. Different methodologies for rendering and simulation are introduced for modeling hair based on the tradeoff between realism and efficiency. In this section, we only focus on the most relevant works in interactive hair simulation and existing mechanics model for strand dynamics and other aspects of hair rendering and simulation can be referred to [15, 16].

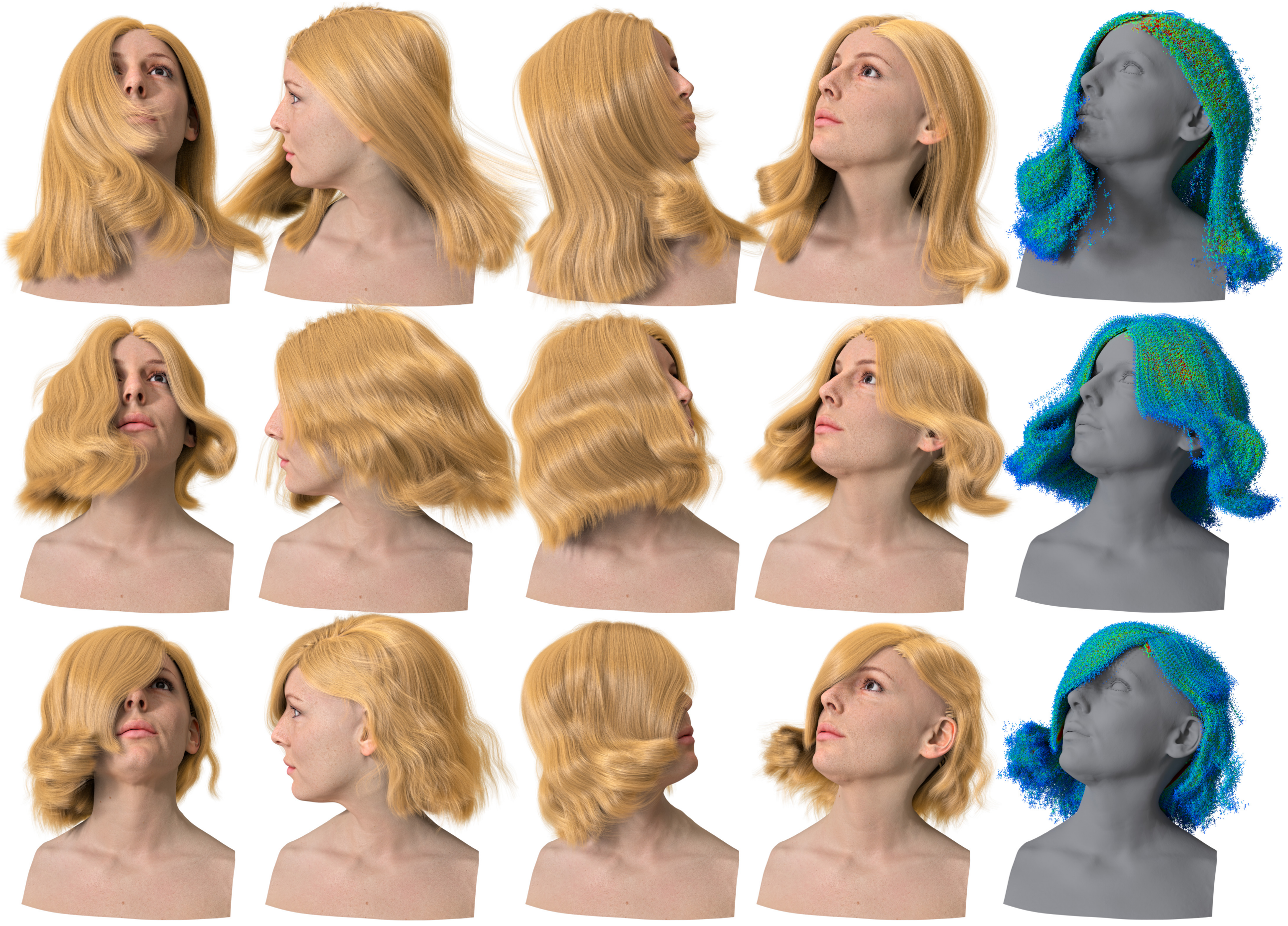

Figure 1: Results from simulating 50K hairs in real time. **First-Fourth**: final rendering results with various hair styles. **Last**: volumetric control using position-based density control, where relative densities are shown with different colors.

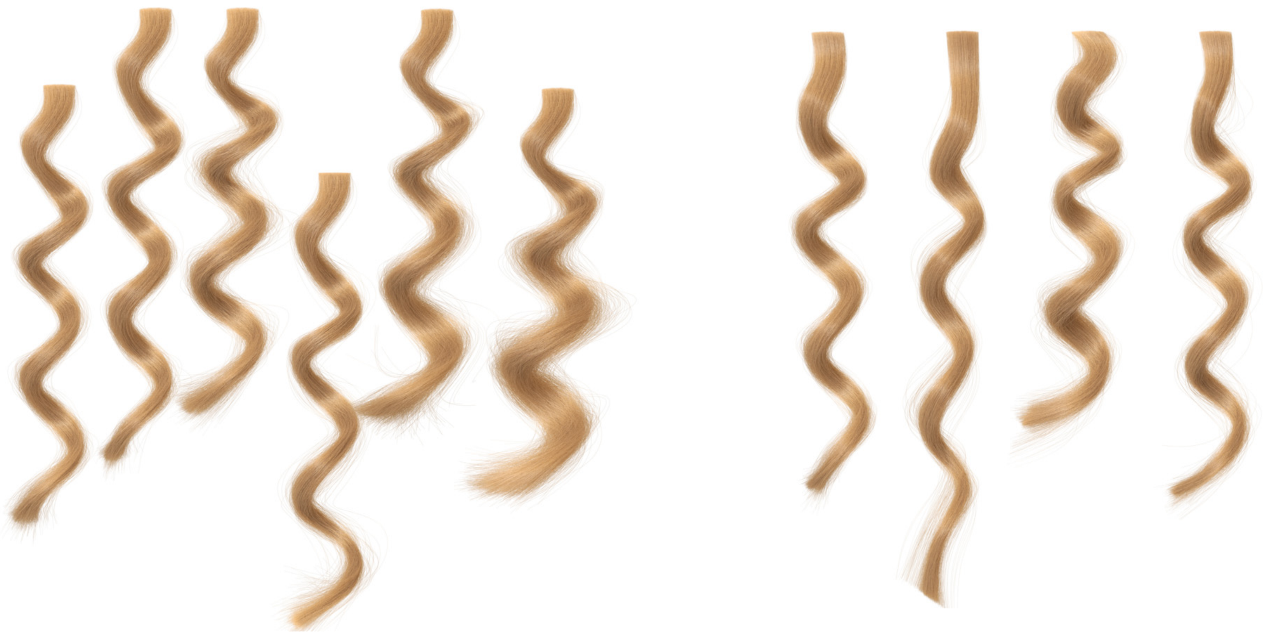

Figure 2: Simulation of a wisp of hair. **Left**: visualization of strand dynamics under severe movements in vertical direction. **Right**: visualization of how hairs deform and restore when drag forces are applied and released at the tips.

2.1. Real-Time Hair Simulation

Compared with offline simulations, real-time hair simulations must utilize a tradeoff between accuracy and time constraint. The main goal of real-time hair simulation is to achieve plausible visual effects while ensuring the computation time satisfies a limited time budget. As a result, early works focused on lowering the DoFs for simulation. A typical scheme for such simplification is the clumped hair model, which models a group of hairs with a single wisp or strip [17, 18, 19, 20, 21, 22]. The hair geometries in clumped models are represented by several disconnected strips that represent a group of clumped hairs with similar motion tendencies. Strands are only textured on the strips; therefore the dynamics of a wisp of hair is simplified to strip motion. External collisions and contacts are often detected and responded to at the strip level. These models significantly lower the DoFs of the hair geometry and are able to perform simulations at low computational complexity. However, such clumped phenomena make them unable to capture the details of a single hair strand and cannot provide high-fidelity results, which makes it undesirable for highly dynamic flipping or wind motions. Interpolation approaches can be viewed as a variation of clumped models, where hairs are interpolated based on a set of guide strands [9, 10, 11]. For instance, Chai et al. [10] introduced skinning interpolation with a data-driven approach, where interpolation weights are pre-calculated from an offline training process [10, 11]. However, poor visual results are still detected when more DoFs are needed (e.g., fuzzy hair). The local shape of the interpolated hair has not been proven to be preserved with element-wise weight interpolation, and using implicit centerline for interpolation makes the interpolated hair lose its physical mechanics. Strange effects can also be found when augmenting the interpolated hairs with additional post-processing stages, such as volumetric protection and self collisions.

In recent years, many studies have focused on simplifying the strand dynamics and succeeded in simulating the whole hair geometry in real time [6, 23, 8]. Müller et al. [6] introduced the DFTL (dynamic follow-the-leader) to model single-strand dynamics in real time. The DFTL method first computes the position with a semi-implicit Euler method and fixes with inextensibility constraints with a forward iteration [6]. They updated velocities with a damping term to hide the implicit uneven mass distribution caused by single iteration. Müller et al. [6] only focused on simulating inextensible straight hair, and curliness is framed along the centerline. Han and Harada [23] introduced a similar constraint-based approach that can maintain the shape of the hair strand with Verlet integration. Sanchez-Banderas et al. [8] combined the work of [23] with [6] and proposed a new DFTL (dynamic follow-the-lead) method that can preserve the local shape of the hair. The local shapes are modeled with hard constraints, which makes them unable to provide stretching and twisting behaviors.

2.2. Strand Dynamics

Strand dynamics controls the mechanics of how a single hair strand behaves. A realistic strand dynamics model should be able to capture the bending, torsion, non-stretching and curliness properties. Various strand dynamics models have been well-studied over the past few decades. One of the first attempts to computationally model hair dynamics was presented by [14]. It uses a simplified mass spring model to capture the hair dynamics. Rosenblum et al. [14] modeled every hair strand as a group of discrete particles connected by stiff springs. The major drawback of such method is that it cannot correctly model the torsional rigidity. In addition, the explicit Euler method causes system vibrations under high stiffness configurations. Anjyo et al. [13] modeled hair strands with one-dimensional projective equations. Such method easily captures the inextensibility of hair strands; however, it still suffers from issues when accounting for the torsion stiffness. In addition, adopting a single-pass iteration from root to tip does not allow one to easily consider external constraints.

A generalized method called *Articulated Rigid Body Chain* (ARB) treats individual hair strand as multiple links connected [24, 25]. When coupling ARB with single un-branched open-loop kinematic chain, this method benefits from linear time complexity compared to other models using implicit iteration; however, curliness is not guaranteed, and the method only succeeds in simulating straight hair styles [15]. Hadap [26] extended the ARB to include a tree structure by solving algebraic equations for allowing analytic constraints. However, the model requires a quadratic problem to be solved to handle collision behaviors. Researchers have also considered modeling single-hair mechanics with Kirchhoff's elastic rod model [27, 28]. One of the first approaches was introduced by [29], where a strand is described as a space framed curve attached at each point on the curve. Bertails et al. [30] introduced the "*Super-Helix*" model built upon the Cosserat and Kirchhoff theories of rods. In the "*Super-Helix*" model, a single hair strand is represented as a piecewise helix with a number of DoFs. Such an approach allows one to capture the bending, twisting and curliness in very few DoFs. However, external constraints, such as collisions and contacts, are difficult to address. Bergou et al. [1] described the Kirchhoff model rod in a discretized form, where the local shape is preserved by the material frame with parallel transport. In addition, the discrete elastic rod model has been successfully embedded in other applications or non-physical based frameworks in subsequent works [31, 32, 33].

On the other hand, mass spring systems have been widely adopted for hair, cloth and elastic solid physical simulations [34, 35, 36]. Petrovic et al. [37] modeled the dynamics of key hairs using mass spring systems. Selle et al. [12] presented a simple but efficient mass spring model, in which a strand is composed of edge, bending, torsion and altitude springs, with a post-processing length-preserving stage for retaining incompressibility. Iben et al. [38] adopted mass spring model in material frame framework, where an explicit centerline is computed with smooth kernel along the curve; the method introduces core springs to maintain the curliness during violent motions.

Previous models mainly adopted implicit Euler method and its variations to resolve the strand dynamics time integrations. Several techniques or simplifications have been used for accelerating the implicit integration. Selle et al. [12] adopted a linearization for the mass spring system by fixing the normalized direction in the integration to reduce the partial differential equation (PDE) into a linear problem. However, this simplification does not seem promising for other high-order non-linear systems like [1, 39]. In addition, Liu et al. [40] embedded mass spring systems into position based framework using block coordinate descent, therein showing a significant speed improvement compared to the original Newton's method. Michels et al. [41] proposed a stiffly accurate integrator based on mathematical reformulation of differential equations, which significantly increases the speed on high stiffness systems.

2.3. Hair Contact and Collision

Interactions among strands are another important factor for achieving realistic hair animations. Friction and static charge adhesion forces cause hair to stick together, demonstrating a tendency to maintain hair style during motion. Intuitively, many approaches treat hair interactions with a segment-wise view. For example, penalty force impulse schemes have been adopted to model stiction and repulsive effects. The interaction forces are dynamically created or removed under certain conditions [12, 25, 38, 42, 43, 44]. Bertails-Descoubes et al. [45] captured dry friction contacts implicitly by treating them as a zero-finding problem of a non-smooth function. In addition, Kaufman et al. [4] showed that the strand-based contact with adaptive high order treatment could correctly simulate high fidelity as well as tangled effects.

Another perspective on self-interaction handling is treating hairs in a volumetric manner. The volumetric way for self interactions normally provides good coarse behaviors, while details are eliminated. Previous method, such as [37], handled interactions with velocity smoothing in voxel grids. Other fluid continuum treatments can be found in [24, 46]. As a recent work, McAdams et al. [47] suggested projecting velocities into a divergence-free form using FLIP to provide a coarse overview and adding details through Lagrangian collision handling. A similar hybrid Eulerian/Lagrangian approach can be found in [48] for capturing collisions for cloth, knit and hair frictional contacts. For interactive interaction modeling, Müller et al. [6] adopted a similar Eulerian method from [37] by averaging the velocities in a background grid. Repulsion effects are simulated by forwarding velocities toward the normalized gradient of the density field [6]. However, the details of collisions are lost since the interactions are only resolved in volume view. Position-based schemes were also widely used for large-scale particle interaction and solid object handling [49]. Chai et al. [10] used coherent link update to maintain penalty forces for fast updating and treated self-collisions as a position-based energy minimization quadratic problem with Cholesky update. But the overall penalty equation and penalty forces are designed to maintain the uniform distribution of hair particles. It still leaves the detailed collisions unsolved [10, 11].

3. Overview

Algorithm 1 describes the time integration steps from t^n to t^{n+1}. $\boldsymbol{X}^n, \boldsymbol{V}^n, \boldsymbol{X}^{n+1}$, and $\boldsymbol{V}^{n+1}$ denote the position and velocity vectors in t^n and t^{n+1}. The body forces at time step t^n are denoted as $\boldsymbol{F}^n$. $\boldsymbol{X}^n, \boldsymbol{V}^n, \boldsymbol{X}^{n+1}, \boldsymbol{V}^{n+1}$, and $\boldsymbol{F}^n \in \mathbb{R}^{3m}$, where m is the number of particles in the system. The integration starts with parallel strand dynamics integration (see Section 4). A nested iteration is then applied for integrating hair dynamics to ensure stability. Each iteration updates the velocities and positions using implicit linear spring approach (see Section 4.2). The positions are further modified with an additional post-processing stage method for ensuring inextensibility (see Section 4.4) and density control (see Section 5.1). $\boldsymbol{V}^{n+1}$ is set to the velocities that achieve the final integrated positions. Finally, we provide a two-stage hair-solid collision test by first fixing the velocities with correct viscosity handling. We commit the final positions with another hair-solid collision detection and response stage (see Section 5.3).

4. Strand Dynamics

4.1. Selle Mass Spring Model

In this paper, we focus on providing a real-time integration method using Selle mass spring model [12]. We briefly review this model while referring readers to the original paper for more details. In Selle mass spring model, a hair strand is discretized

Procedure 1 Time Integration

```
1: $\Delta t = \frac{1}{\text{MaxNestedIteration}}\left(t^{n+1} - t^n\right)$
2: $\boldsymbol{V}_0^{n+1} = \boldsymbol{V}^n$
3: $\boldsymbol{X}_0^{n+1} = \boldsymbol{X}^n$
4: for $i = 1$ , $i \leq$ MaxNestedIteration, ++$i$ do
5:     $\boldsymbol{V}_i^{n+1} = \mathcal{V}\left(\boldsymbol{X}_{i-1}^{n+1}, \boldsymbol{V}_{i-1}^{n+1}, \boldsymbol{F}^n + \boldsymbol{F}_{\text{collision}}^{n+1}, \Delta t\right)$
6:     $\boldsymbol{X}_i^{n+1} = \boldsymbol{X}_{i-1}^{n+1} + \Delta t \boldsymbol{V}_i^{n+1}$
7: end for
8: $\boldsymbol{V}_i^{n+1} = \mathcal{V}\left(\boldsymbol{X}_{i-1}^{n+1}, \boldsymbol{V}_{i-1}^{n+1}, \boldsymbol{F}^n, \Delta t\right)$
9: $\boldsymbol{X}_i^{n+1} = \boldsymbol{X}_{i-1}^{n+1} + \Delta t \boldsymbol{V}_i^{n+1}$
10: Apply constraint to $\boldsymbol{V}_i^{n+1}$, $\boldsymbol{X}_i^{n+1}$
11: Self Interaction Handling:
12: Modify $\boldsymbol{X}^{n+1}$ and with volume preservation
13: Modify $\boldsymbol{V}^{n+1}$ with viscosity
14: Modify $\boldsymbol{X}^{n+1}$, $\boldsymbol{V}^{n+1}$ with detailed collisions
15: Solid Collision (for particles inside solid object):
16: $\boldsymbol{V}^{n+1} = \left(\boldsymbol{X}^{n+1} - \boldsymbol{X}^n\right) \Big/ \left(t^{n+1} - t^n\right)$
17: Fix $\boldsymbol{V}^{n+1}$ with hair-solid collisions
18: $\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \left(t^{n+1} - t^n\right) \boldsymbol{V}^{n+1}$
19: Fix $\boldsymbol{X}^{n+1}$ with hair-solid collisions
```

as particles connected with springs to simulate the stretching, bending and torsion behaviors for hair strands. Edge springs are connected to adjacent particles, thereby forming the skeleton of the hair strand. Bending springs are attached between every other particle. The edge and bending springs form triangles, which represent the orientation of the hair. The torsion springs (or twist springs) are added to form tetrahedrons in order to keep the hair strand from collapsing. To prevent degeneration, altitude springs are attached for each tetrahedron formed by 4 adjacent particles. For co-linear segments, the tetrahedrons are degenerate and unable to present orientation. One virtual particle is interpolated for each co-linear edge to avoid collapse. Figure 3 shows how various hair styles are represented in the Selle mass spring model.

4.2. Linearization

Let $\boldsymbol{X}_i^n, \boldsymbol{X}_i^{n+1}, \boldsymbol{V}_i^n, \boldsymbol{V}_i^{n+1} \in \mathbb{R}^{3m_i}$ denote the position and velocity vector of strand i with particle size m_i. The backward Euler equations from time t^n to t^{n+1} can be written as $\boldsymbol{X}_i^{n+1} = \boldsymbol{X}_i^n + \Delta t \boldsymbol{V}_i^{n+1}$ and $\boldsymbol{V}_i^{n+1} = \boldsymbol{V}_i^n + \Delta t \boldsymbol{M}^{-1}\left(\boldsymbol{F}_i^n + \boldsymbol{S}_i^{n+1} - \boldsymbol{G}\boldsymbol{V}_i^{n+1}\right)$, where $\Delta t = t^{n+1} - t^n$, $\boldsymbol{M}, \boldsymbol{G} \in \mathbb{R}^{3m_i \times 3m_i}$ are the mass matrix and damping coefficient matrix, respectively $\boldsymbol{F}_i^n$ denotes the external force vector for strand i, and $\boldsymbol{S}_i^{n+1}$ is the internal spring force vector in t^{n+1}. For a particle-wise representation, we can rewrite $\boldsymbol{X}_i^n$ as $\boldsymbol{X}_i^n = \left[(\boldsymbol{x}_1^n)^T, \cdots, (\boldsymbol{x}_{m_i}^n)^T\right]^T$, where $\boldsymbol{x}_i^n \in \mathbb{R}^3$ denotes the particle's position with local index i. Alternatively, we can define $\boldsymbol{V}_i^n = \left[(\boldsymbol{v}_1^n)^T, \cdots, (\boldsymbol{v}_{m_i}^n)^T\right]^T$, $\boldsymbol{S}_i^n = \left[(\boldsymbol{s}_1^n)^T, \cdots, (\boldsymbol{s}_{m_i}^n)^T\right]^T$, where $\boldsymbol{v}_i^n$ and $\boldsymbol{s}_i^n$ denote the velocity and sum of the spring forces of a particle with local index i. $\boldsymbol{s}_i^n$ can be expressed as $\boldsymbol{s}_i^{n+1} = \sum_{j \in P(i)} k_{i,j}\left(\left(\boldsymbol{x}_j^{n+1} - \boldsymbol{x}_i^{n+1}\right)^T \hat{\boldsymbol{d}}_{i,j}^{n+1} - l_{i,j}\right)\hat{\boldsymbol{d}}_{i,j}^{n+1}$, where $k_{i,j}$ and $l_{i,j}$ are the stiffness and rest length for the spring connected particle i and j. $P(j)$ denotes the particle index set which for particle i with internal springs, and $\hat{\boldsymbol{d}}_{i,j}^{n+1}$ is the normalized direction from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$: $\hat{\boldsymbol{d}}_{i,j}^{n+1} = \left(\boldsymbol{x}_j^{n+1} - \boldsymbol{x}_i^{n+1}\right) \Big/ \|\boldsymbol{x}_j^{n+1} - \boldsymbol{x}_i^{n+1}\|$. Selle et al. [12] suggested fixing $\hat{\boldsymbol{d}}^{n+1}$ to $\hat{\boldsymbol{d}}^n$ to make the integration linear, in which we obtain:

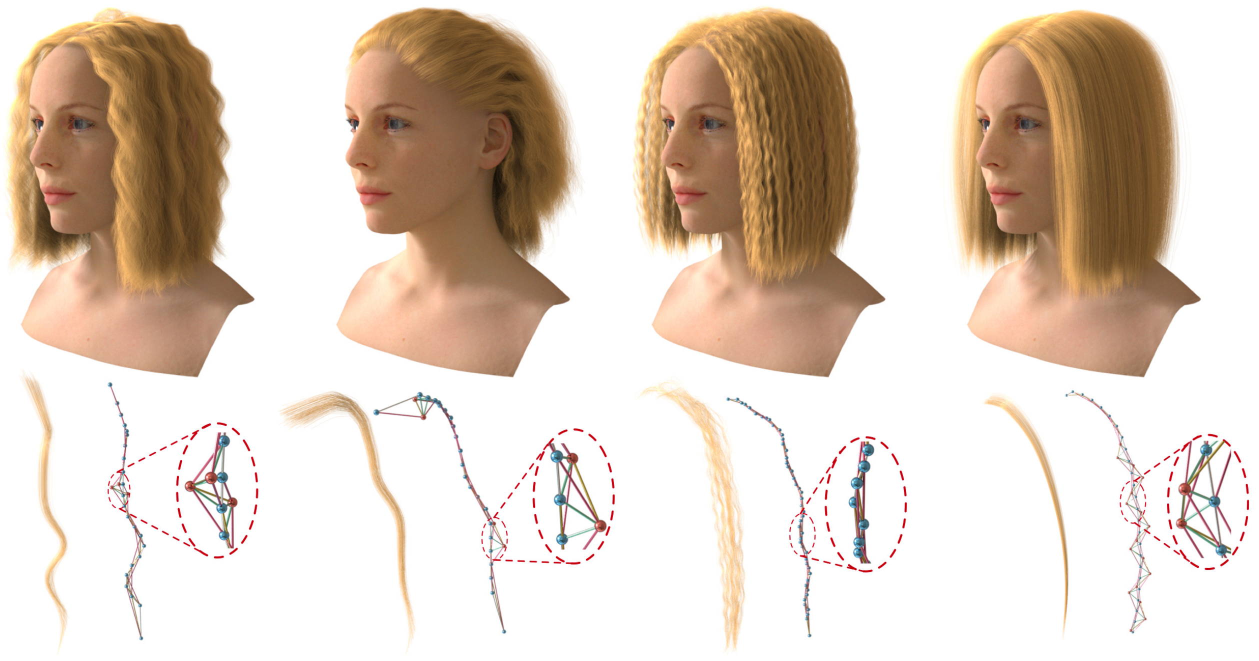

Figure 3: Different hair styles modeled by Selle mass spring model. The top row shows 4 different hair styles (wavy, curly, helical and straight). The bottom row shows a wisp of the corresponding hair geometry and its mass spring structure. A hair strand can be represented as a connected structure of normal particles (red sphere), virtual particles (blue sphere), edge springs (white edge), bending springs (yellow edge), torsion springs (magenta edge) and extra edge springs (green edge).

$$\begin{aligned} \boldsymbol{s}_i^{n+1} = &\sum_{j \in P(i)} k_{i,j} \left(\|\boldsymbol{x}_j^n - \boldsymbol{x}_i^n\| - l_{i,j} \right) \hat{\boldsymbol{d}}_{i,j}^n \\ &+ \sum_{j \in P(i)} k_{i,j} \Delta t \boldsymbol{D}_{i,j}^n \left(\boldsymbol{v}_j^{n+1} - \boldsymbol{v}_i^{n+1} \right) \end{aligned} \tag{1}$$

where $l_{i,j}$ denotes the rest length for the corresponding spring, and

$$\boldsymbol{D}_{i,j}^n = \left(\hat{\boldsymbol{d}}_{i,j}^n \right)^T \hat{\boldsymbol{d}}_{i,j}^n, \boldsymbol{D}_{i,j}^n \in \mathbb{R}^{3\times 3}$$

is the direction matrix for $\boldsymbol{x}_i^n$ and $\boldsymbol{x}_j^n$. Equation 1 indicates that the spring force vector $\boldsymbol{S}_i^{n+1}$ can be represented as

$$\boldsymbol{S}_i^{n+1} = \hat{\boldsymbol{S}}_i^n + \Delta t \boldsymbol{C}_i^n \boldsymbol{V}_i^{n+1} \tag{2}$$

where $\hat{\boldsymbol{S}}_i^n$ is the spring force vector with $\boldsymbol{X}^n$ and $\boldsymbol{C}_i^n = \left[\boldsymbol{c}_{i,j}^n \right]_{m_i \times m_i}$ is the connectivity matrix, $\boldsymbol{C}_i^n \in \mathbb{R}^{3m_i \times 3m_i}$, and $\boldsymbol{c}_{i,j}^n \in \mathbb{R}^{3\times 3}$ is deduced from Equation 1 as:

$$\boldsymbol{c}_{i,j}^n = \begin{cases} k_{i,j} \boldsymbol{D}_{i,j}^n & j \in P(i) \\ -\sum\limits_{j \in P(i)} k_{i,j} \boldsymbol{D}_{i,j}^n & i = j \\ \boldsymbol{O_{3\times 3}} & \text{otherwise} \end{cases} \tag{3}$$

Substituting Equation 2 into implicit Euler equations yields the linear equation:

$$\begin{aligned}&\left(\boldsymbol{I}_{3m_i\times 3m_i}+\Delta t\boldsymbol{M}^{-1}\boldsymbol{G}-\Delta t^2\boldsymbol{M}^{-1}\boldsymbol{C}_i^n\right)\boldsymbol{V}_i^{n+1}\\&=\boldsymbol{V}_i^n+\Delta t\boldsymbol{M}^{-1}\left(\boldsymbol{F}_i^n+\hat{\boldsymbol{S}}_i^n\right)\end{aligned}\tag{4}$$

4.3. Fast Integration

By utilizing the 1D local connectivity structure, we could provide a must faster decomposition strategy. A particle with index i that only has finite connections with its neighbors implies that the matrix for the implicit equation is banded. For a band matrix, the LU decomposition can be exactly resolved using only two iterations (i.e. a forward and a backward sweep). For example, [50] adopts such strategy for computing the articulated-body inertias. [51] also utilizes a similar two-pass recursive computation for integrating the "Super-Helices" rod model. In addition, [52] shows that by adopting a combined reduced & maximal coordinate formulation we could achieve near linear time complexity even when implicit elasticity is added between arbitrary bodies rather than only between immediate neighbors. In this section, we show how this strategy could be used for resolving the implicit integration for the Selle mass spring system.

For clarity, the time step superscript n is omitted in the following discussion and we reuse the index i to denote the local index for particles. Traditionally, iteration-based methods such as PCG are unable to satisfy real-time requirements and needs multiple iterations to converge.

Thanks to the local connectivity in the mode of [12], a particle with local index i only has numerical terms from $i-3$ to $i+3$ within the same strand, making $\boldsymbol{A}$ a heptadiagonal band matrix in $\mathbb{R}^3$. Consider the linear equation in 4 $\boldsymbol{AV}=\boldsymbol{b}$, $\boldsymbol{A}=[\boldsymbol{a}_{i,j}]_{m_i\times m_i}$ and $\boldsymbol{b}=[\boldsymbol{b}_i]_{m_i}$, $\boldsymbol{a}_{i,j}\in\mathbb{R}^3$, $\boldsymbol{b}_i\in\mathbb{R}^3$. $\boldsymbol{a}_{i,j}$ and $\boldsymbol{b}_i$ are represented as:

$$\begin{aligned}&\boldsymbol{a}_{i,j}=\begin{cases}-\Delta t^2\mu k_{i,j}\boldsymbol{D}_{i,j} & |i-j|=1,2,3\\ (1+\Delta t\mu g)\,\boldsymbol{I}_{3\times 3}+\Delta t^2\mu\sum\limits_{j\in P(i)}k_{i,j}\boldsymbol{D}_{i,j} & i=j\\ \boldsymbol{O}_{3\times 3} & \text{otherwise}\end{cases}\\&\boldsymbol{b}_i=\boldsymbol{v}_i+\Delta t\mu\left(\boldsymbol{f}_i+\hat{\boldsymbol{s}}_i\right)\end{aligned}\tag{5}$$

where μ is the inverse of particle mass and g is the damping coefficient. The heptadiagonal band matrix linear equation in $\mathbb{R}^3$ can be exactly solved with only a forward and another backward sweep. The first sweep tries to decompose $\boldsymbol{A}=\boldsymbol{LU}$ and computes the intermediate result $\boldsymbol{V}'$, where $\boldsymbol{L},\boldsymbol{U}\in\mathbb{R}^{3m_i\times 3m_i}$, $\boldsymbol{u}_{i,j}\in\mathbb{R}^{3\times 3}$, $\boldsymbol{u}_{i,i}=\boldsymbol{I}_{3\times 3}$ for $j\geq i$ and $\boldsymbol{l}_{i,j}\in\mathbb{R}^{3\times 3}$for $i\geq j$. The heptadiagonal property implies that $\boldsymbol{L}$ and $\boldsymbol{U}$ will have at most 4 continuous non-zero $\mathbb{R}^{3\times 3}$ block matrices. Therefore, $\boldsymbol{l}_{i,j}=\boldsymbol{O}_{3\times 3}$ when $i-j<3$ and $\boldsymbol{u}_{i,j}=\boldsymbol{O}_{3\times 3}$ when $j-i>3$. For other cases, $\boldsymbol{l}_{i,j}$ and $\boldsymbol{u}_{i,j}$ can be directly computed during the first sweep: $\boldsymbol{l}_{i,j}=\boldsymbol{a}_{i,j}-\sum\limits_{k=\max(1,i-3)}^{j-1}\boldsymbol{l}_{i,k}\boldsymbol{u}_{k,j}$ and $\boldsymbol{u}_{i,j}=\left(\boldsymbol{l}_{i,i}\right)^{-1}\boldsymbol{a}_{i,j}-\left(\boldsymbol{l}_{i,i}\right)^{-1}\sum\limits_{k=\max(1,j-3)}^{i-1}\boldsymbol{l}_{i,k}\boldsymbol{u}_{k,j}$. We compute the intermediate results along with the decomposition. Let $\boldsymbol{V}'=[\boldsymbol{v}_i']_{m_i}$, $\boldsymbol{V}'\in\mathbb{R}^{3m_i}$, $\boldsymbol{v}_i'\in\mathbb{R}^3$ be intermediate vector of $\boldsymbol{V}'=\boldsymbol{L}^{-1}\boldsymbol{b}$. We have $\boldsymbol{v}_i'=\boldsymbol{l}_{i,i}^{-1}\left(\boldsymbol{b}_i-\sum\limits_{j=\max(0,i-3)}^{i-1}\boldsymbol{l}_{i,j}\boldsymbol{v}_j'\right)$. Another backward sweep yields the final result

$\boldsymbol{U}\boldsymbol{V} = \boldsymbol{V}'$, which indicates $\boldsymbol{v}_i = \boldsymbol{v}'_i - \sum_{j=i+1}^{\min(i+3,m)} \boldsymbol{u}_{i,j}\boldsymbol{v}_j$. Note that the altitude spring in [12] does not break the above assumption, because all possible edge/edge or point/face connections can only exist inside tetrahedrons. The fast integration method makes the computation time of implicit integration only proportional to the number of particles. Only trivial matrix operations are involved in the integration step, suggesting that the speed can be further improved by using SSE and AVX instructions. Algorithm 2 shows the ppseudocode of the forward and backward iteration of the strand dynamics integration. 190

Procedure 2 Strand Dynamics Integration

```
Input: $\boldsymbol{A} = [\boldsymbol{a}_{i,j}]_{m\times m}, \boldsymbol{b} = [\boldsymbol{b}_i]_m$
Output: $\boldsymbol{V} = [v_i]_m$
1: for $i = 1, i \leq m,$ ++$i$ do
2:     for $j = \max(i-3, 0), j \leq i,$ ++$j$ do
3:         $\boldsymbol{l}_{i,j} = \boldsymbol{a}_{i,j}$
4:         for $k = \max(1, i-3), k \leq j-1,$ ++$k$ do
5:             $\boldsymbol{l}_{i,j} = \boldsymbol{l}_{i,j} - \boldsymbol{l}_{i,k}\boldsymbol{u}_{k,j}$
6:         end for
7:     end for
8:     $\boldsymbol{v}'_i = \boldsymbol{b}_i$
9:     for $j = \max(i-3, 0), j \leq i-1,$ ++$j$ do
10:        $\boldsymbol{v}'_i = \boldsymbol{v}'_i - \boldsymbol{l}_{i,j}\boldsymbol{v}'_j$
11:    end for
12:    $\boldsymbol{v}'_i = \boldsymbol{l}_{i,i}^{-1}\boldsymbol{v}'_i$
13: end for
14: for $i = m, i \geq 1,$ –$i$ do
15:     for $j = i+1, j \leq \min(i+3, M),$ ++$j$ do
16:         $\boldsymbol{u}_{i,j} = \boldsymbol{a}_{i,j}$
17:         for $k = \max(1, j-3), k \leq i-1,$ ++$k$ do
18:             $\boldsymbol{u}_{i,j} = \boldsymbol{u}_{i,j} - \boldsymbol{l}_{i,k}\boldsymbol{u}_{k,j}$
19:         end for
20:         $\boldsymbol{u}_{i,j} = \boldsymbol{l}_{i,i}^{-1}\boldsymbol{u}_{i,j}$
21:     end for
22:     $\boldsymbol{v}_i = \boldsymbol{v}'_i$
23:     for $j = i = 1, j \leq \min(i+3, M),$ ++$j$ do
24:         $\boldsymbol{v}_i = \boldsymbol{v}_i - \boldsymbol{u}_{i,j}\boldsymbol{v}_j$
25:     end for
26: end for
27: return $\boldsymbol{V} = [v_i]_m$
```

4.4. *Inextensibility Preservation*

Inextensibility can be enforced by increasing the stiffness of stretching springs, although such an approach vibrates the integration. In addition, some authors have suggested the use of constraint-based models as a post-processing stage to protect the inextensibility behavior and ensure the integration remains unchanged [12, 1]. The original model adopts a biased strain limiting process by marching from the root to tip and projecting over-stretching segments to desired positions. Using a single 195 forward iteration implicitly assumes that the particles near the tip have a substantially smaller mass compared to the previous particles, producing an overestimation of the positions of the particles near the tip. Such an artifact can be easily found when coupling with thin object collision (see Figure 4). Ensuring inextensibility could be viewed as a problem of satisfying several distance constraints while minimizing the position distortion, making it an ideal model under position based dynamics

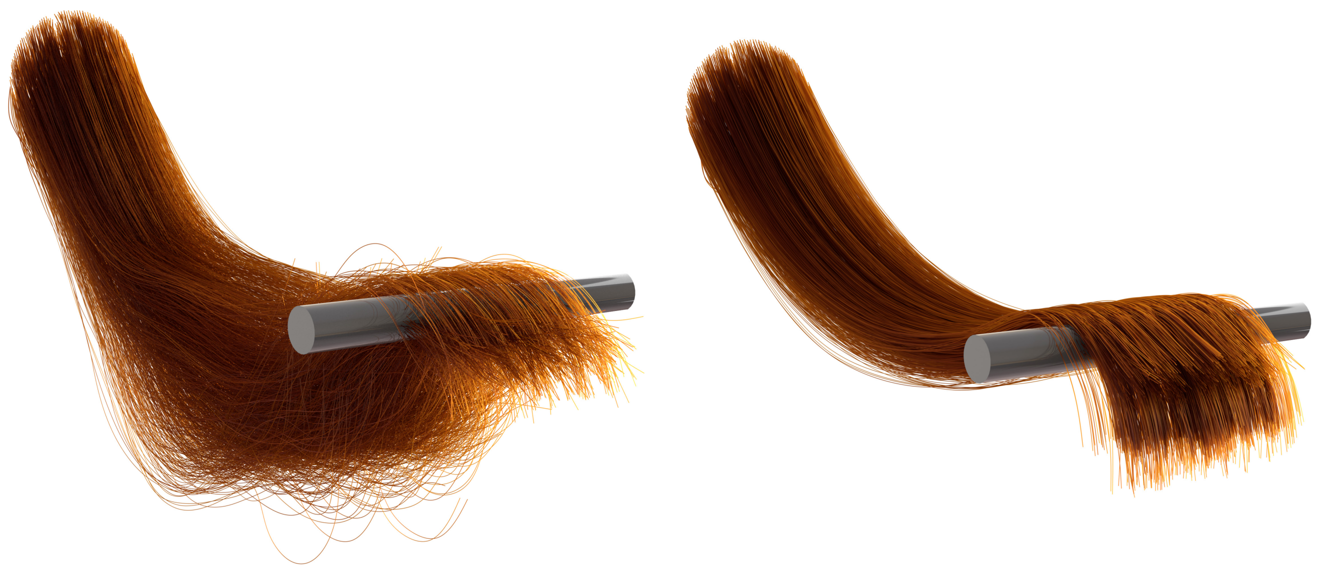

Figure 4: Coupling strain limiting with a thin object with relatively high static friction. **Left**: strain limiting using only a single forward iteration, where the particle positions are overestimated at the tips, causing a failure of the simulation for collision response. **Right**: strain limiting with position based length preservation.

description. Practically, two or three iterations work well for most cases, and weak compression or extension for edges still yield plausible results.

5. Hair Interactions

Self interactions usually dominate the fidelity when a large assembly of hairs is simulated. Although handling interactions only with collision detection and response is sufficient to provide enough detail, it still requires the integration system to address the highly nonlinearity implicit problem [4]. Such integration takes minutes to hours to complete a single frame in collision-intensive simulation cases. [47] suggests adopting a hybrid Eulerian/Lagrangian method can benefit from the efficiency of volume solver and details are also well preserved by the Lagrangian approach [47], while the original method that adopts FLIP and full Lagrangian checking makes heavy computation burden for interactive usage. Our method resembles the same methodology described in [47] for handling self interactions with two individual stages. The first stage treats hair geometry in a continuum manner, while another segment-wise collision handling stage is introduced for maintaining details for collisions. We accelerate the volume preservation with parallel position based density control and reuse the collision pairs for detailed collision handling. This ensures that both methods are able to complete in interactive time budget.

5.1. Volume Preservation

Similar to other SPH methods used in fluid simulation, we discretize the hair geometry into a set of particles with radius r_0 and unit mass. Incompressibility of particles is enforced by adding density constraints for particle i, where $C_i = (r_0)^3 \sum_{j \in \mathcal{N}_i} W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) - 1$. $W : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^+$ is the filter kernel, h is the kernel radius and $\mathbb{N}_i$ is the particle set with the kernel radius. Normally $h = 3r_0$ works well for most test cases. The goal is to satisfy all C_i constraints

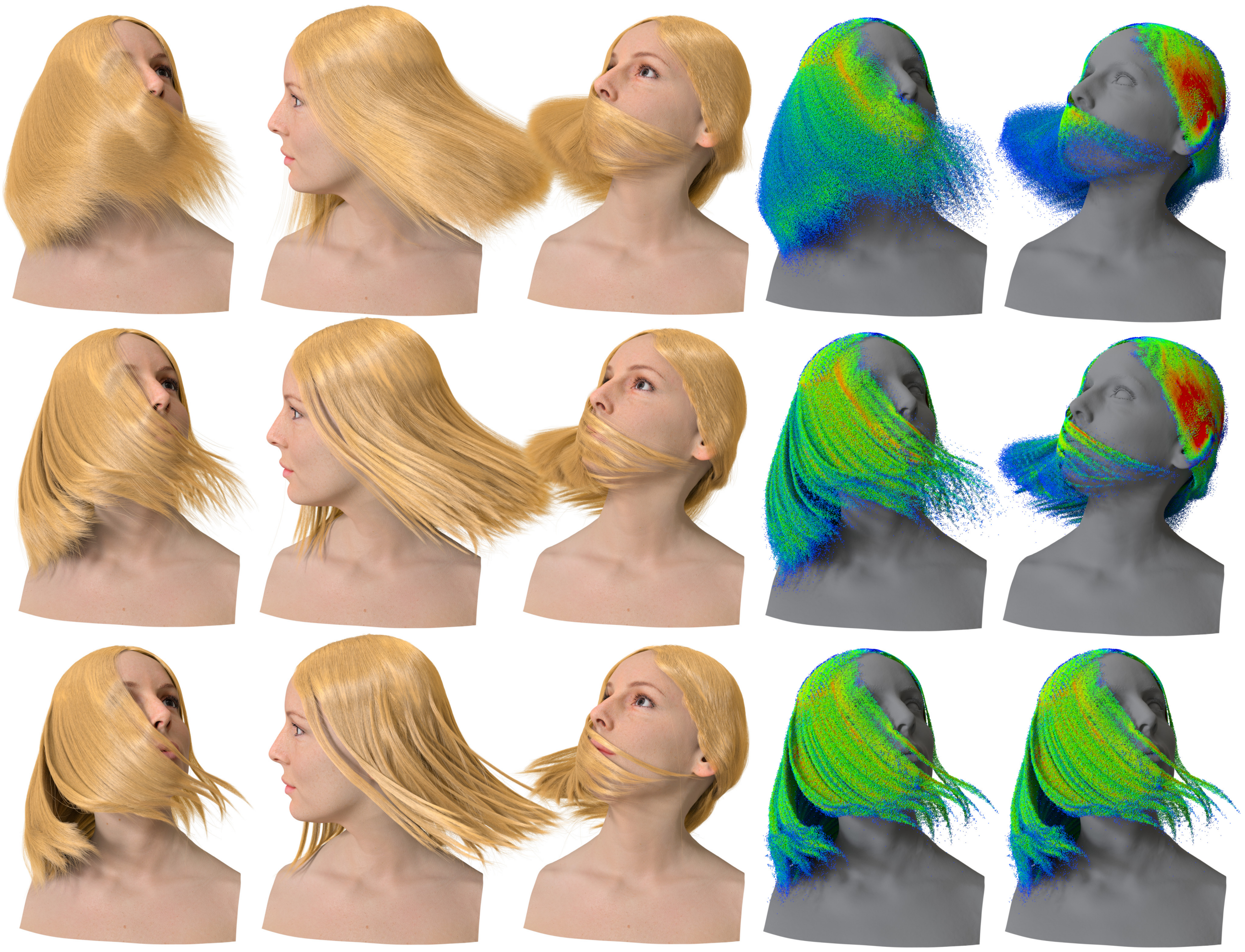

Figure 5: Various clumping effects by tuning viscosity c and constraint relaxation parameter ϵ. Increasing viscosity or reducing ϵ causes strands to have a tendency to clump together.

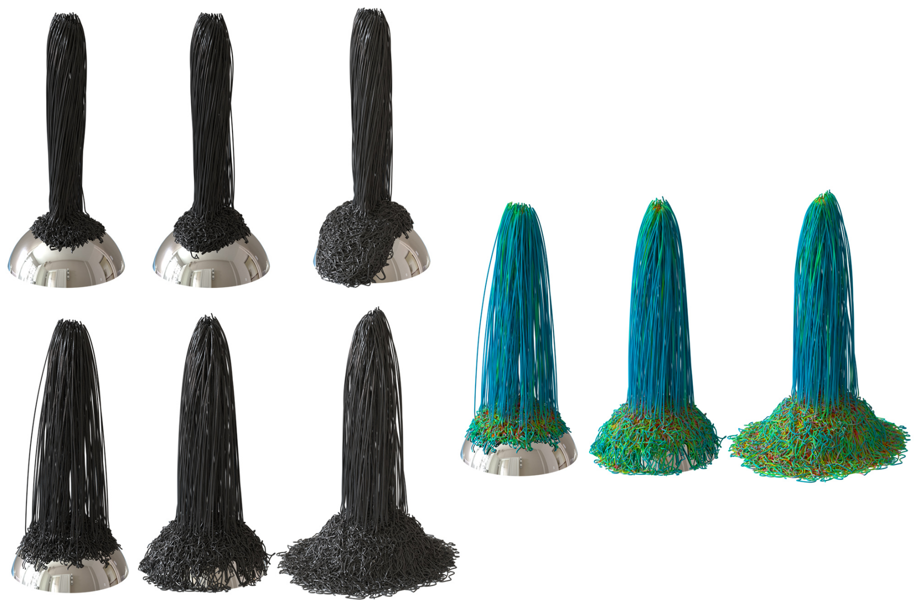

Figure 6: Cascaded noodles example simulated by volumetric preservation. **Left**: Simulation results of how noodles cascade down and collide due to external collision and self interactions. The top figure shows the result simulated with high viscosity, causing noodles have a tendency to clump due to relatively high internal viscosity. The bottom figure shows how noodles react in low viscosity coefficient condition. **Right**: Density visualization (colored from blue to red) for the low viscosity test case.

with $C_i = 0$. The overall update in positions is the accumulated result for each constraint with single Newton iteration $\frac{-\Delta \boldsymbol{x}_i}{\nabla W(\boldsymbol{x}_i - \boldsymbol{x}_j, h)(r_0)^3} = \sum_{j \in \mathcal{N}_i} \left(\frac{C_i}{\|\nabla C_i\|^2 + \epsilon} + \frac{C_j}{\|\nabla C_j\|^2 - \epsilon} + s_{\text{corr}} \right)$.

The s_{corr} term is used to avoid tensile instability, and we refer readers to [53] for more implementation details. Compared with the original method that uses ϵ to avoid instability when the gradient approaches 0, we choose a kernel where the gradient norm $\|\nabla W\|$ is large in its domain so that the gradient norm $\|\nabla C_i\| \to 0$ will never occur. We still keep ϵ as an extra parameter to soften the position constraints. We tune ϵ based on whether the local density is compressing ($C_i < 0$) so that we can soften the clumping effects while maintaining the incompressibility as stiff as possible (see Figure 5).

The clumping can be tuned by adjusting various viscosity coefficients. To do this, we modify the velocity of particle i with smoothing kernel

$$\boldsymbol{v}_i' = \boldsymbol{v}_i + c \sum_{j \in \mathcal{N}_i} (\boldsymbol{v}_j - \boldsymbol{v}_i)\, W(\boldsymbol{p}_j - \boldsymbol{p}_i, h)$$

, where c is the viscosity coefficient. For efficiency, we reuse the neighborhood relation $\mathcal{N}_i$ in density estimation to avoid additional re-computation for the spatial hashing structure. Figure 6 shows a comparison of the simulation results by tuning the viscosity for modeling clumping.

The method proposed in [53] is used to provide a divergence-free form for fluid dynamics; thus, every particle does not have additional connections with others in the SPH framework. This assumption does not hold for hair simulation and results in a situation in which the shape of strands are deformed after PBD corrections. To correctly provide shape preservation in this stage, we add high stiff stretching, bending and torsion constraints as well as the density constraints into the PBD solver, forcing the final results to converge without shape collapse or deformation.

We note that previous works, such as [10], also adopt the PBD method for hair correction. The method used in [10] tends to maintain a uniform distance for every interesting particle pair but minimizes the position change using a blend between two cost functions. However, the description of the cost function implies that each particle pair is arbitrarily pushed in the original direction without correct consideration of local density. In addition, treating two constraints with the blend method cannot ensure that they are both satisfied.

5.2. Detailed Collisions

The volume preservation stage provides a roughly visual approximation for hair geometry in a continuum way, but the coarse overview cannot correctly handle localized collision cases. We further modify the hair geometry with Lagrangian collision handling step with impulse strategy. For efficiency, we limit the maximum number of collided edges for each hair segment, which allows us to quit in a reasonable time even for collision intensive test cases. Intuitively, two edges possibly collide if an actual collision occurs in the previous frame, which indicates us to reuse the collision edge buffer. For each segment, our method first prunes the invalid collided edges in the buffer and appends new ones if the number of collisions does not exceed. Because the previous volume preservation provides a precondition for detailed collision stage, effectiveness is guaranteed compared to other simplex segment collision handling strategies. Figure 7 shows the comparison results for a collision-intensive test case with or without detailed the collision stage.

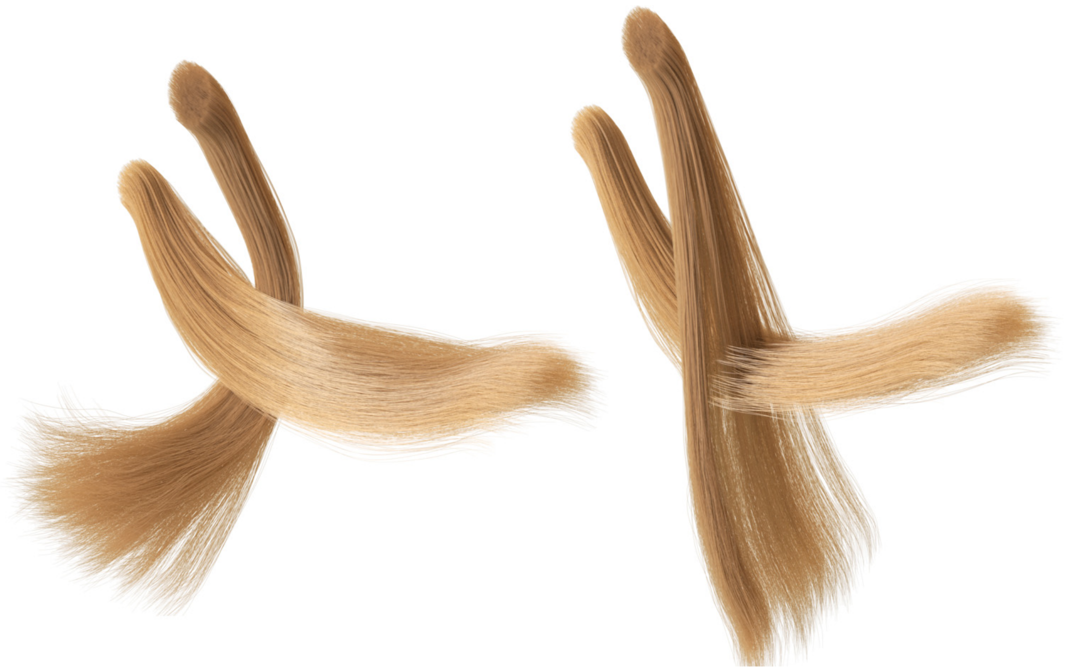

Figure 7: Collision test with two wisps of hair. **Left**: full self interactions method, including detailed collision; **Right**: volume preservation only.

5.3. Solid Collision

Updating particle positions before solid collision can create incorrect friction response behaviors. Consider a rigid body with infinite friction coefficient; any ill-considered position update before an actual collision will cause an additional tangential velocity along the surface, resulting in an undesirable position update for that surface. We adopt a two-stage solid collision response model to address this issue with correct friction contact modeling. For the particles inside the rigid body, instead of committing the particle positions, we set the velocities to the targeted positions before the solid collision: $\boldsymbol{v}^{n+1} = \left(\boldsymbol{x}^{n+1} - \boldsymbol{x}^n\right) \Big/ \left(t^{n+1} - t^n\right)$. In the first stage, $\boldsymbol{v}^{n+1}$ is modified with friction for solid objects. We recompute the predicted positions from the modified velocities. Any positions failing in the secondary collision detection test will be re-updated by being pushed to the boundary.

We build the level-set lattice on the fly [54]. Let $\sigma_p : \mathbb{R}^3 \to \mathbb{R}$ be the signed distance function, and $\sigma_v : \mathbb{R}^3 \to \mathbb{R}^3$ denotes the velocity of the solid object in $\mathbb{R}^3$. For any particle with invalid boundary condition $\sigma_p(\boldsymbol{x} + \Delta t \boldsymbol{v}) < 0$, the velocity is fixed by the following equation: $\boldsymbol{v}' = \boldsymbol{u} + \max\left(0, 1 - \mu \frac{|\boldsymbol{v}_N - \boldsymbol{u}_N|}{|\boldsymbol{v}_T - \boldsymbol{u}_T|}\right)(\boldsymbol{v}_T - \boldsymbol{u}_T)$,where $\boldsymbol{u} = \sigma_v(\boldsymbol{x} + \Delta t \boldsymbol{v})$ and v_N, u_N and v_T, u_T are the projected and tangential velocities on the level-set normal $\nabla \sigma_p(\boldsymbol{x} + \Delta t \boldsymbol{v})$. If the targeted position $\boldsymbol{x} + \Delta t \boldsymbol{v}'$ is still inside the solid object, it will be further pushed in the secondary stage: $\boldsymbol{x}' = \boldsymbol{x} + \left(\frac{\nabla \sigma_p}{\|\nabla \sigma_p\|} \sigma_p\right)(\boldsymbol{x} + \Delta t \boldsymbol{v}')$.

6. Implementation Details and Comparison

6.1. Implementation Details

We test our model with different parameters and scenes. The "Head Rotation" scene is composed of 3 head movements and 3 hair styles. All hair geometries contain 50,000pm/m^2 strands, and each strand is discretized into 25 particles. For

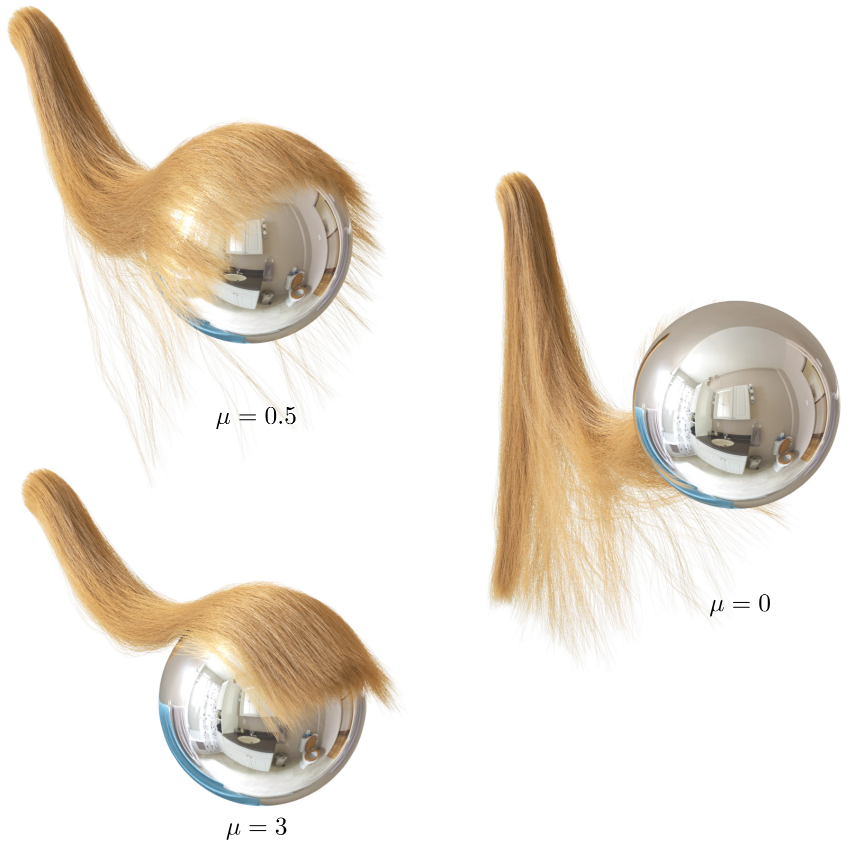

Figure 8: Visualization of solid collisions with correct friction handling using two-stage strategy.

the "Head Rotation" scene (see Figure 1), the stretching stiffness is set to 20,000pm/m^2 and the stiffnesses of other types of springs are set to 10,000pm/m^2, where the term pm denotes the average mass for each particle. Such setting ensures that the tetrahedron will not collapse and provide sufficient stiffness for inextensibility preservation in the integration step. For length preservation post processing, we set our length tolerance coefficient to 0.5%. We also limit the maximum PBD iteration times to 5 for timing consideration. For two stage collision strategy, we accelerate our neighborhood querying processing by creating a spatial hashing structure on the fly, which is adopted in [55]. The grid size is set to 1.5 times the kernel radius. We set the kernel radius to 0.003m for the volumetric stage and the radius to 0.004 for detailed collision handling. The friction is robustly set to 1.2 for all "Head Rotation" test cases. For signed distance computation, we discretize the mesh with a resolution of $64 \times 64 \times 64$ in a pre-allocated memory space, and the hair-solid friction coefficient is set to 0.8 for all test cases. The "Strand Dynamics Drag Force" (see Figure 2) demonstrates the scenario in which the tips are stretched with external forces. The "Strand Dynamics Motion" (see Figure 2) shows the response under severe movements. The setup resembles the previous "Head Rotation" configuration, except that the stiffness is set to 50,000pm/m^2 for edge springs and 25,000pm/m^2 for bending and torsion springs. We increase the kernel radius to 0.006m for volumetric preservation and the radius to 0.008 for detailed collisions. For "Body Collision" (see Figure 8) and "Self Collision" (see Figure 7), we adopt the same parameter configuration used in "Strand Dynamics". The geometry settings and detailed computation time of different substeps are shown in Table 1. For the overall computation time, our method succeeds in completing full simulations with over

Scene	**P**	**S**	***SDI***	***CSR***	***SC***	***BC***
HR1-S1	1.25M	50K	10ms	11ms	25ms	1ms
HR1-S2	1.25M	50K	10ms	12ms	27ms	1ms
HR1-S1	1.25M	50K	10ms	12ms	26ms	1ms
HR2-S1	1.25M	50K	10ms	13ms	21ms	1ms
HR2-S2	1.25M	50K	11ms	12ms	26ms	1ms
HR2-S3	1.25M	50K	11ms	11ms	30ms	1ms
HR3-S1	1.25M	50K	11ms	12ms	20ms	1ms
HR3-S2	1.25M	50K	10ms	12ms	22ms	1ms
HR3-S3	1.25M	50K	10ms	12ms	26ms	1ms
SDM(Figure 2)	24K	1.8K	297us	8ms	9ms	-
SDDF(Figure 2)	24K	1.8K	256us	5ms	7ms	15us
BC(Figure 8)	65K	2.5K	1ms	1ms	1ms	25us
SC(Figure 7)	130K	5K	1ms	1ms	11ms	-

Table 1: The overall simulation time for each integration is decomposed into *SDI* (strand dynamics integration), *CSR* (coarse stiction and repulsion), *SC* (self collision) and *BC* (body collision). We use "P" for particle number in the scene, "S" to denote number of strands in the scene. We use short name notation for scene in the table, where "HRx-Sy" denotes "Head Rotation x - Scene y"; SDM denotes "Strand Dynamics Motion"; "SDDF" denotes "Strand Dynamics Drag Force"; "BC" denotes "Body Collision" and "SC" denotes "Self Collision".

	PCG-D	**PCG-IC**	**PCG-LQ**	**BiCGSTAB**	**LU**	**QR**	**CO**	**LLT**	**LDLT**	**SVD-J**	**SVD-BDC**	**Ours(CPU)**
2500×15	19.489	46.501	89.661	24.157	53.636	78.699	44.856	12.903	13.001	196.388	191.326	**0.573**
2500×50	57.805	103.957	405.604	73.071	323.413	389.229	198.141	41.511	16.043	876.985	924.494	**1.252**
10000×25	78.934	181.998	354.237	97.119	206.982	283.319	174.238	48.211	49.59	744.219	762.094	**2.950**
10000×50	222.552	373.001	1588.237	279.284	1232.351	1496.336	731.369	162.868	58.233	3714.323	3569.531	**5.432**
50000×25	688.654	1436.885	3428.574	791.358	3220.224	4311.271	4033.267	526.764	274.333	10192.033	10387.094	**18.826**

Table 2: The strand dynamics computation time with different internal linear problem solvers: PCG-D (PCG with diagonal preconditioner), PCG-IG (PCG with incomplete Cholesky preconditioner), PCG-LQ (PCG with least square precondition), BiCGSTAB (iterative stabilized bi-conjugate gradient), LU (standard LU decomposition), QR (QR decomposition), CO (complete orthogonal decomposition), LLT (standard Cholesky decomposition), LDLT (Cholesky decomposition with pivoting), SVD-J(two-sided Jacobi SVD decomposition), SVD-BDC (bidiagonal divide and conquer SVD decomposition) and our fast decomposition method (measured in *ms*).

1M particles within 100*ms* in the "Head Rotation" scene. In most simulation scenes, approximately 60% of the computation budget is taken by self-collision detection and response caused by creating the spatial structure and more complicated checking strategies; the remainder consists of the hair integration (20%) and volume protection (20%). For collision-intensive case such as Figure 7, the occupancy of self collisions increases dramatically due to the massive bounding box overlapping. The overall pipeline is implemented in CUDA and tested with a NVIDIA GTX TITAN graphics card.

6.2. *Comparison*

For strand dynamics integration, we compare our fast decomposition method with standard PCG solvers. In Figure 9 (left), we show 4 contours demonstrating the integration time for the global PCG method (CPU), the parallel PCG method (CPU) and our method (CPU and GPU). The parallel PCG method adopts the same decoupling schemes introduced in Section 3 while replacing the linear equation solver with PCG (diagonal precondition). Our method (CPU implementation) is 150x faster than the global PCG (CPU) baseline and is also 7-10x faster than the parallel PCG (CPU). Our GPU implementation can achieve another 10x speed improvement to our CPU implementation. We further compare our fast decomposition method with other factorization based (standard LU decomposition, QR decomposition, complete orthogonal decomposition, LLT, LDLT, SVD-Jacobi and SVD-BDC) and iterative based (CG with diagonal precondition, CG with incomplete Cholesky precondition, least square CG and stabilized bi-CG) linear problem solvers. The results (see Table 2) show that our method achieves one order

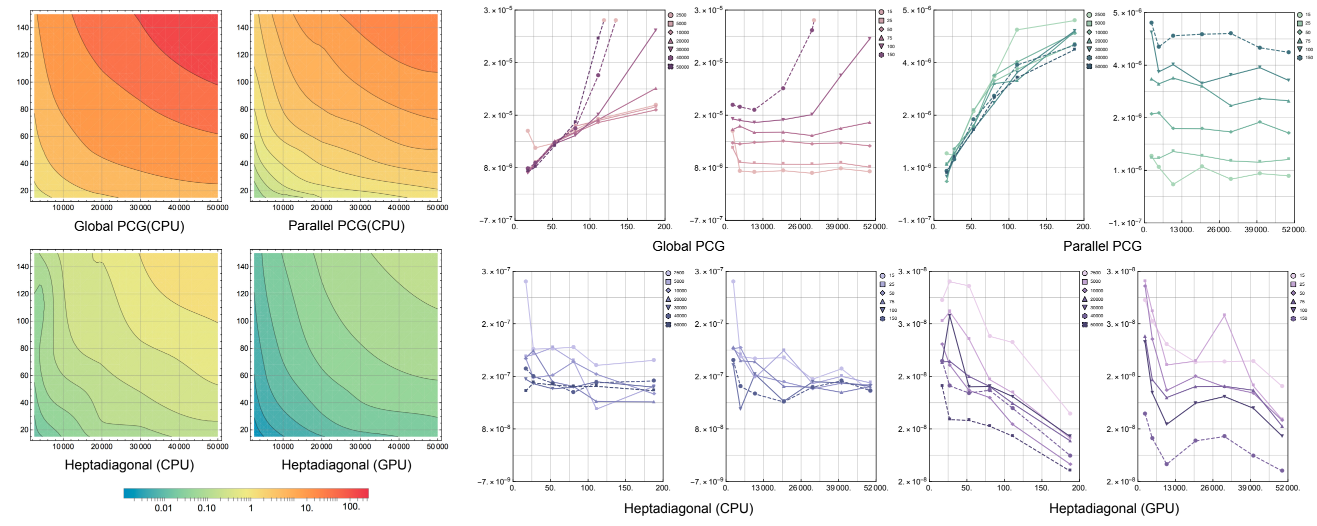

Figure 9: **Left**: The contour plot of the integration computation time for different strand dynamics methods: global PCG (CPU), parallel PCG (CPU) and heptadiagonal decomposition on both CPU and GPU. The horizontal axis denotes the strand number, and the vertical axis denotes the average particle number in a single strand. Time is measured in seconds (s). **Right**: The average computation time for single particle with different particle and strand settings for global PCG, parallel PCG, heptadiagonal decomposition on CPU and heptadiagonal decomposition on GPU. In the left column, the right axis denotes the strand number. In the right column, the right axis denotes the average particle number for a hair strand.

of speed improvement compared to the fastest solver in all test cases. For scalability, we further test the average computation speed for each particle with different strand and discretization settings. We adopt 6 particle discretization configurations (15, 25, 50, 75, 100 and 150 per strand) and 7 strand configurations (2000, 5000, 10000, 20000, 30000, 40000 and 50000 per geometry). Figure 9 (right) shows that increasing the discretization yields a significant speed decrease for the PCG solvers. On the other hand, our method maintains a similar speed tendency for all the geometry configurations, which indicates that the overall computation time for our method keeps a linear relationship only with the particle number and is robust to other factors. In our GPU implementation, we observe that the average speed increases when more particles are added to the system due to a better occupancy to the CUDA cores.

7. Conclusion

We propose an integration model for interactively capturing the rich behaviors of hair mechanics on the fly, making it an ideal model for interactive usage. For strand dynamics integration, we accelerate the Selle mass spring representation by adopting a fast decomposition scheme. Such method allows us to solve the implicit linear problem using only a forward and backward sweep iterations. Inspired by the density control method used in fluid simulations, we adopt the PBD scheme to correctly simulate volumetric effects within a reasonable time budget, and details are further preserved with a post-processing Lagrangian collision stage. The whole pipeline can be fully integrated on GPU and shows a significant speed improvement compared to other existing methods. There exists several aspects that can be further improved upon as our future works. The torsion stiffness is still expressed in particle positions. We will maintain the efficiency while investigating a more elegant geometric representation to correctly model torsion in a frame-based schemes. In addition, we will resolve the time-dependent stiffness problem in volumetric density control, which is caused by the original artifact of PBD framework.

Declaration of Competing Interest

There are no conflicts of interest to declare for this manuscript.

Acknowledgements

This work was supported in part by the National High Technology Research and Development Program of China (863 Program) under Grant 2017YFB1002701, in part by the National Natural Science Foundation of China under Grant 61872241, Grant 61672502, and Grant 61632003, in part by The Hong Kong Polytechnic University under Grant P0030419 and Grant P0030929, and in part by the University of Macau under Grant MYRG2019-00006-FST.

[1] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, E. Grinspun, Discrete elastic rods, ACM transactions on graphics (TOG) 27 (3) (2008) 63. 1, 5, 10

[2] G. Daviet, F. Bertails-Descoubes, L. Boissieux, A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics, in: ACM Transactions on Graphics (TOG), Vol. 30, ACM, 2011, p. 139. 1

[3] A. Derouet-Jourdan, F. Bertails-Descoubes, G. Daviet, J. Thollot, Inverse dynamic hair modeling with frictional contact, ACM Transactions on Graphics (TOG) 32 (6) (2013) 159. 1

[4] D. M. Kaufman, R. Tamstorf, B. Smith, J.-M. Aubry, E. Grinspun, Adaptive nonlinearity for collisions in complex rod assemblies, ACM Transactions on Graphics (TOG) 33 (4) (2014) 123. 1, 6, 11

[5] G. Gornowicz, S. Borac, Efficient and stable approach to elasticity and collisions for hair animation, in: Proceedings of the 2015 Symposium on Digital Production, ACM, 2015, pp. 41–49. 1

[6] M. Müller, T.-Y. Kim, N. Chentanez, Fast simulation of inextensible hair and fur., VRIPHYS 12 (2012) 39–44. 1, 4, 6

[7] T. Martin, W. Engel, N. Thibieroz, J. Yang, J. Lacroix, Tressfx: Advanced real-time hair rendering, GPU Pro 5 (2014) 193–209. 1

[8] R. M. Sanchez-Banderas, H. Barreiro, I. Garca-Fernndez, M. Prez, Real-time inextensible hair with volume and shape, CEIG. 1, 4

[9] P. Guan, L. Sigal, V. Reznitskaya, J. K. Hodgins, Multi-linear data-driven dynamic hair model with efficient hair-body collision handling, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2012, pp. 295–304. 1, 4

[10] M. Chai, C. Zheng, K. Zhou, A reduced model for interactive hairs, ACM Transactions on Graphics (TOG) 33 (4) (2014) 124. 1, 4, 6, 13

[11] M. Chai, C. Zheng, K. Zhou, Adaptive skinning for interactive hair-solid simulation, IEEE transactions on visualization and computer graphics 23 (7) (2017) 1725–1738. 1, 4, 6

[12] A. Selle, M. Lentine, R. Fedkiw, A mass spring model for hair simulation, ACM Transactions on Graphics (TOG) 27 (3) (2008) 64. 2, 5, 6, 7, 9, 10

[13] K.-i. Anjyo, Y. Usami, T. Kurihara, A simple method for extracting the natural beauty of hair, in: ACM SIGGRAPH Computer Graphics, Vol. 26, ACM, 1992, pp. 111–120. 2, 5

[14] R. E. Rosenblum, W. E. Carlson, E. Tripp, Simulating the structure and dynamics of human hair: modelling, rendering and animation, Computer Animation and Virtual Worlds 2 (4) (1991) 141–148. 2, 4

[15] K. Ward, F. Bertails, T.-Y. Kim, S. R. Marschner, M.-P. Cani, M. C. Lin, A survey on hair modeling: Styling, simulation, and rendering, IEEE transactions on visualization and computer graphics 13 (2) (2007) 213–234. 2, 5

[16] F. Bertails, S. Hadap, M.-P. Cani, M. Lin, T.-Y. Kim, S. Marschner, K. Ward, Z. Kačić-Alesić, Realistic hair simulation: animation and rendering, in: ACM SIGGRAPH 2008 classes, ACM, 2008, p. 89. 2

[17] C. K. Koh, Z. Huang, A simple physics model to animate human hair modeled in 2d strips in real time, Proceedings of the Eurographic workshop on Computer animation and simulation (2001) 127–138. 4

[18] E. Sugisaki, Y. Yu, K. Anjyo, S. Morishima, Simulation-based cartoon hair animation, in: 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2005, WSCG'2005 - In Co-operation with EUROGRAPHICS, 2005, pp. 117–122. 4

[19] C. K. Koh, Z. Huang, Real-time animation of human hair modeled in strips (2000) 101–110. 4

[20] Y. Guang, H. Zhiyong, A method of human short hair modeling and real time animation, in: 10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings., 2002, pp. 435–438. 4

[21] H. D. Taskiran, U. Gudukbay, Physically-based simulation of hair strips in real-time., in: WSCG (Short Papers), 2005, pp. 153–156. 4

[22] W. Liang, Z. Huang, An enhanced framework for real-time hair animation, in: 11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings., 2003, pp. 467–471. 4

[23] D. Han, T. Harada, Real-time hair simulation with efficient hair style preservation, VRIPHYS (2012) 45–51. 4

[24] S. Hadap, N. Magnenat-Thalmann, Modeling dynamic hair as a continuum, in: Computer Graphics Forum, Vol. 20, Wiley Online Library, 2001, pp. 329–338. 5, 6

[25] J. T. Chang, J. Jin, Y. Yu, A practical model for hair mutual interactions, in: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM, 2002, pp. 73–80. 5, 6

[26] S. Hadap, Oriented strands: dynamics of stiff multi-body system, in: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association, 2006, pp. 91–100. 5

[27] M. Grégoire, E. Schömer, Interactive simulation of one-dimensional flexible parts, Computer-Aided Design 39 (8) (2007) 694–707. 5

[28] D. Metaxas, J. Popovic, Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. 5

[29] D. K. Pai, Strands: Interactive simulation of thin solids using cosserat models, in: Computer Graphics Forum, Vol. 21, Wiley Online Library, 2002, pp. 347–352. 5

[30] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, J.-L. Lévêque, Super-helices for predicting the dynamics of natural hair, in: ACM Transactions on Graphics (TOG), Vol. 25, ACM, 2006, pp. 1180–1187. 5

[31] N. Umetani, R. Schmidt, J. Stam, Position-based elastic rods, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2014, pp. 21–30. 5

[32] T. Kugelstadt, E. Schömer, Position and orientation based cosserat rods, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2016, pp. 169–178. 5

[33] Y. R. Fei, H. T. Maia, C. Batty, C. Zheng, E. Grinspun, A multi-scale model for simulating liquid-hair interactions, ACM Transactions on Graphics (TOG) 36 (4) (2017) 56. 5

[34] R. Bridson, R. Fedkiw, J. Anderson, Robust treatment of collisions, contact and friction for cloth animation, ACM Transactions on Graphics (ToG) 21 (3) (2002) 594–603. 5

[35] K.-J. Choi, H.-S. Ko, Stable but responsive cloth, in: ACM SIGGRAPH 2005 Courses, ACM, 2005, p. 1. 5

[36] M. Teschner, B. Heidelberger, M. Muller, M. Gross, A versatile and robust model for geometrically complex deformable solids, in: Computer Graphics International, 2004. Proceedings, IEEE, 2004, pp. 312–319. 5

[37] L. Petrovic, M. Henne, J. Anderson, Volumetric methods for simulation and rendering of hair, Pixar Animation Studios 2 (4). 5, 6

[38] H. Iben, M. Meyer, L. Petrovic, O. Soares, J. Anderson, A. Witkin, Artistic simulation of curly hair, in: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ACM, 2013, pp. 63–71. 5, 6

[39] M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, E. Grinspun, Discrete viscous threads, in: ACM Transactions on Graphics (TOG), Vol. 29, ACM, 2010, p. 116. 5

[40] T. Liu, A. W. Bargteil, J. F. O'Brien, L. Kavan, Fast simulation of mass-spring systems, international conference on computer graphics and interactive techniques 32 (6) (2013) 214. 5

[41] D. L. Michels, V. T. Luan, M. Tokman, A stiffly accurate integrator for elastodynamic problems, ACM Transactions on Graphics (TOG) 36 (4) (2017) 116. 5

[42] S. Jimenez, A. Luciani, Animation of interacting objects with collisions and prolonged contacts, in: Modeling in Computer Graphics, Springer, 1993, pp. 129–141. 6

[43] R. Bridson, S. Marino, R. Fedkiw, Simulation of clothing with folds and wrinkles, in: ACM SIGGRAPH 2005 Courses, ACM, 2005, p. 3. 6

[44] B. Criswell, K. Derlich, D. Hatch, Davy jones' beard: rigid tentacle simulation, in: ACM SIGGRAPH 2006 Sketches, ACM, 2006, p. 117. 6

[45] F. Bertails-Descoubes, F. Cadoux, G. Daviet, V. Acary, A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies, ACM Transactions on Graphics (TOG) 30 (1) (2011) 6. 6

[46] Y. Bando, B.-Y. Chen, T. Nishita, Animating hair with loosely connected particles, in: Computer Graphics Forum, Vol. 22, Wiley Online Library, 2003, pp. 411–418. 6

[47] A. McAdams, A. Selle, K. Ward, E. Sifakis, J. Teran, Detail preserving continuum simulation of straight hair, in: ACM Transactions on Graphics (TOG), Vol. 28, ACM, 2009, p. 62. 6, 11

[48] C. Jiang, T. Gast, J. Teran, Anisotropic elastoplasticity for cloth, knit and hair frictional contact, ACM Transactions on Graphics (TOG) 36 (4) (2017) 152. 6

[49] J. Bender, M. Müller, M. A. Otaduy, M. Teschner, M. Macklin, A survey on position-based simulation methods in computer graphics, Computer Graphics Forum 33 (6) (2014) 228–251. 6

[50] R. Featherstone, The calculation of robot dynamics using articulated-body inertias, The International Journal of Robotics Research 2 (1) (1983) 13–30. 9

[51] F. Bertails, Linear time super-helices, in: Computer graphics forum, Vol. 28, Wiley Online Library, 2009, pp. 417–426. 9

[52] Y. Wang, N. J. Weidner, M. A. Baxter, Y. Hwang, D. M. Kaufman, S. Sueda, Redmax: efficient & flexible approach for articulated dynamics, ACM Transactions on Graphics (TOG) 38 (4) (2019) 1–10. 9

[53] M. Macklin, M. Müller, Position based fluids, ACM Transactions on Graphics (TOG) 32 (4) (2013) 104. 13

[54] K. Erleben, H. Dohlmann, Signed distance fields using single-pass gpu scan conversion of tetrahedra, in: Gpu Gems 3, Addison-Wesley, 2008, pp. 741–763. 14

[55] M. Ihmsen, N. Akinci, M. Becker, M. Teschner, A parallel sph implementation on multi-core cpus, in: Computer Graphics Forum, Vol. 30, Wiley Online Library, 2011, pp. 99–112. 15