Interactive Hair Simulation on the GPU using ADMM

GILLES DAVIET, NVIDIA, France

Fig. 1. Real-time physically-based editing session of a 86,000 curves groom at 10% simulated guides, using the Discrete Elastic Rods model with Coulomb friction. From left to right: input groom; sagged pose; hair selection and manipulation; trimming the selection; same actions on the screen-left side; end result.

We devise a local–global solver dedicated to the simulation of Discrete Elastic Rods (DER) with Coulomb friction that can fully leverage the massively parallel compute capabilities of moderns GPUs. We verify that our simulator can reproduce analytical results on recently published cantilever, bend–twist, and stick–slip experiments, while drastically decreasing iteration times for high-resolution hair simulations. Being able to handle contacting assemblies of several thousand elastic rods in real-time, our fast solver paves the ways for new workflows such as interactive physics-based editing of digital grooms.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Massively parallel algorithms**.

Additional Key Words and Phrases: hair simulation

ACM Reference Format:
Gilles Daviet. 2023. Interactive Hair Simulation on the GPU using ADMM. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 6–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3588432.3591551

1 INTRODUCTION

Hair is a complex medium; while the elastic nature of individual strands can in itself exhibit interesting shapes and dynamics, the overall behavior is also largely driven by the numerous frictional interactions between them, resulting in considerable difficulties for numerical simulators.

Another challenge is digital grooming, the art of designing high-quality hair strands for virtual characters. The overwhelming majority or digital grooms are created through purely geometric manipulation of curves, at the high-end trying to match photographic references over several view angles. However, since the resulting grooms contain no embedded physical information, their dynamical behavior may end-up differing significantly from the artist's intent; in particular, the addition of gravity in the simulation may result in an undesired sagging effect. Several works have tried to address this issue by attempting to recover rest pose and/or physical parameters from the geometry [Derouet-Jourdan et al. 2013; Hsu et al. 2022; Iben et al. 2019; Lesser et al. 2022]. However, this problem is both under- and over- constrained; under-constrained, because for any given geometry there are several mathematically viable ways to achieve equilibrium, balancing the relative effect of elasticity and contact forces; and over-constrained, because this technique typically assumes that the artist-provided geometry is correct; however, the layering and density of strands in the input groom may not actually allow for a physically sensible solution.

Rather than trying to infer physical quantities *a posteriori*, we can instead ensure the grooming process respect physical constraints by manipulating rods inside a fully-fledged simulation. This way, the grooming output contains not only geometrical but also rest pose information, leading to predictable dynamic behavior. However, such a workflow was until now limited by the computational times required for high-fidelity hair simulations. In the following, we describe a solver that can take advantage of the computing power of modern GPUs and take first steps towards an actually interactive physics-based groom editing process.

2 RELATED WORK

Many efforts have been dedicated to the animation of hair, and several real-time techniques have emerged [Kmoch et al. 2010; Oshita 2007; Wu and Yuksel 2016]. While our goal is also to design an efficient solver, here we relax the real-time constraint, and instead target physical realism by focusing on rod and contact models rich enough to reproduce the experiments of Romero et al. [2021].

Cosserat and Kirchhoff rods. As hair strands typically feature a length L much larger than their mean radius R, slender rod theories such as the Kirchhoff (which accounts for only bending and twisting) and Cosserat (which adds stretching and shearing) models have emerged as popular ways of describing their behavior. Both models equip the curve with a material frame encoding its local orientation.

Pai [2002] first introduced the Cosserat model to Computer Graphics, using the relative material frame transform at vertices as degrees of freedom. Bertails et al. [2006] then Casati and Bertails-Descoubes [2013] proposed the "Super" curvature-based discretizations of the Kirchhoff model, using piecewise-constant and piecewise-linear elements, respectively. Spillmann and Teschner [2007] proposed a maximal–coordinates discretization of Cosserat rods, with degrees

SIGGRAPH '23 Conference Proceedings, August 6–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08.
https://doi.org/10.1145/3588432.3591551

of freedom for both the vertex positions of piecewise-linear centerline and per-edge material frame orientation. Bergou et al. [2010, 2008] kept the piecewise-linear centerline representation, but focused on the shear-free Kirchhoff model by limiting the material frame degrees of freedom to a single per-edge scalar twist angle. This yields the DER model that we will use in our solver, though we will opt for the deformation measure proposed by Korner et al. [2021], based on the re-scaling of the rotation vector between the material frames of adjacent edges, rather than the original deformation measure from Bergou et al. [2008], based on the projection of the curvature binormal onto an "average" material frame.

Recently, Romero et al. [2021] validated that both the "Super" and DER discretizations of the Kirchhoff model are not only able to reproduce analytical results, but also match physical experiments.

Local and hybrid integrators. Due to the high stiffness of hair strands, implicit time integration is generally necessary to achieve stable results, historically using a global Newton-like algorithm. Reduced-coordinates models benefit from the linearity of the elastic responses in their degrees of freedom but suffer from a dense stiffness matrix, while maximal-coordinates models require more iterations with larger and sparser left-hand sides. None perform well on GPUs due to sub-optimal memory access patterns and poor conditioning often requiring double precision solves.

In contrast, the XPBD [Macklin et al. 2016] implicit time integrator is purely local, and Kugelstadt and Schömer [2016] adapted the Cosserat discretization from [Spillmann and Teschner 2007] to the position-based setting. However, while XPBD be can efficiently parallelized on GPUs for strand-like topologies using constraint coloring, it suffers from slower convergence on long chains of constraints; strains are only propagated to the direct neighbors at each iteration, and techniques like follow-the-leader [Müller et al. 2012] or long–range constraints Müller et al. [2017] do not apply for the bending and twisting terms. Deul et al. [2018] improved convergence by incorporating direct solve, while Macklin et al. [2019] argue for aggressive substepping.

The Projective Dynamics [PD; Bouaziz et al. 2014] global–local integrator aims for a middle ground, with with an embarrassingly parallel local step, and a global strain-propagating step that can be solved efficiently. Although the PD framework does not allow for arbitrary energies, Soler et al. [2018] successfully implemented the Cosserat model, and [Ly et al. 2020] demonstrated support for hard contacts with Coulomb friction. Narain et al. [2016] devised a generalization of PD using the Aternating Direction Method of Multipliers (ADMM), from which our solver is derived.

Frictional contacts. Frictional collisions account for a large part of hair appearance and motion and have been the subject of many works, starting from volumetric [Hadap and Magnenat-Thalmann 2001; McAdams et al. 2009; Petrovic et al. 2005] and penalty-based [Choe et al. 2005; Selle et al. 2008] approaches. Daviet et al. [2011] argued for the visual importance of handling Coulomb friction accurately, and devised a complementarity-based solver scaling up to a few thousand strands. Kaufman et al. [2014] adapted their solver to handle the nonlinearities of the DER model and were able to simulate tens of thousands of rods. Daviet [2020] used ADMM to separate the elasticity from the frictional contact complementarity problem, leading to reduced computational and memory costs and allowing for simulation at full human hair density. Our approach for handling frictional contacts builds on theirs, but adapts their mostly sequential solver to the massively parallel GPU architectures.

Following Li et al. [2020] barrier-based techniques have recently gained in popularity for their robustness and strong interpenetration-free guarantees, and variants focusing on codimensional objects [Li et al. 2021] and GPU-friendly [Lan et al. 2022] simulation have emerged. However, the very thin nature of hair strands ($\leq 0.1mm$) and rapidly changing contact topology both contribute to the need for frequent barrier updates, making the performance of these techniques comparatively less attractive for large-scale hair simulations.

3 CONTRIBUTIONS

Our central contribution is a carefully crafted ADMM-based decomposition of the implicit time integration incremental problem for DER with frictional contacts, resulting in massively parallel and single-precision friendly global solve, local elasticity and feasible projection steps (Section 4). We then verify in Section 5.1 that our simulator can reproduce the theoretical results on the cantilever, bend–twist and stick–slip experiments devised by Romero et al. [2021], while also being able to handle full-scale hair simulations orders of magnitude faster than traditional CPU-based solvers (Section 5.2). Finally, in Section 5.3 we lay down the first steps towards a physically-augmented groom editing environment.

4 SIMULATOR

We now briefly recall our elastic rod and frictional contact models, then proceed to describe our ADMM-based implicit time integrator.

4.1 Model

We use the DER [Bergou et al. 2010, 2008] discretization, so that our degrees of freedom are the m centerline vertex positions $(\mathbf{x}_i) \in \mathbb{R}^{3\times m}$ and scalar per-edge twists (θ_j), plus, in the dynamic setting, the corresponding linear and angular velocities $(\mathbf{v}_i)$ and (ω_j). To simplify indexing, we extend the number of twist and angular velocity variables to m by introducing a ghost twist, kinematically fixed to zero, at the end of each rod. We denote $\overline{e_j}$ the rest length of edge j, and $\overline{v_i}$ the Voronoi length associated to vertex i, $\overline{v_i} := \frac{1}{2}\left(\overline{e_i} + \overline{e_{i-1}}\right)$. The rods cross-section is characterized by a mean radius R and an ellipticity coefficient (flatness) ℓ, defined such that the major and minor radii are $(1+\ell)R$ and $(1-\ell)R$, respectively. The cross-section area is then $A = \pi R^2\left(1-\ell^2\right)$, and the moment of inertia around the tangent $I = A\left(1+\ell^2\right)/2$.

Elasticity. The stretching energy associated to an edge j is $\mathcal{E}_{s,j} := \frac{E_s}{2} A\overline{e_j}\varepsilon_j^2$, with E_s the stretch Young modulus and ε the stretching strain. The bending energy associated to an inner vertex i is

$$\mathcal{E}_{b,i} := \frac{E_b}{2}\left(\kappa_i - \bar{\kappa_i}\right)^T \frac{K}{\overline{v_i}}\left(\kappa_i - \bar{\kappa_i}\right),$$
$$K := \frac{AR^2}{4}\,\mathrm{diag}\left(\left(1-\ell\right)^2, \left(1+\ell\right)^2, \frac{1+\ell^2}{1+\nu}\right)$$

with E_b the bending Young modulus, ν the Poisson ratio, and κ (resp. $\bar{\kappa}$) the current (resp. rest) bending strain measure in $\mathbb{R}^3$. To evaluate ε and κ, we can use either the original strain measures from Bergou

et al. [2008] or the alternative ones from [Korner et al. 2021]. The two bending measures are in agreement for small angles ϕ between edges, but differ at large angles; the former scales with $2\tan(\phi/2)$, the latter with $2\sin(\phi/2)$. Moreover, Korner et al. [2021] consider a hyperelastic stretch behavior, $\varepsilon_j := |e_j|^2/\overline{e_j} - \overline{e_j}$, rather than the usual linear measure $\varepsilon_j := |e_j| - \overline{e_j}$. Romero et al. [2021] validated the model from Bergou et al. [2008] against analytic results on their cantilever and bend–twist experiments; we show in Section 5.1 that the model from Korner et al. [2021] also pass these tests. In practice we observe the non-divergent bending energy from Korner et al. [2021] to exhibit swifter convergence, and use it for all our examples.

External forces. Our rods are subjected to an external acceleration field $\mathbf{g}$, and in the dynamic setting, inertia and air drag forces are also considered. The mass lumped to vertex i is $m_i = \rho A\overline{v_i}$, with ρ the volumetric mass of the material, and the rotational inertia associated with edge j is $\rho I\overline{e_j}$. We use a simple linear drag model with dynamic viscosity ξ, justified by the low Reynolds number for flow around hair strands, and leave more complex self-shadowing and coupling effects for future work. Overall, external forces contribute the following per-vertex and per-edge incremental potentials,

$$\mathcal{E}_{e,i} := \frac{1}{2}\rho A\overline{v_i}\left(\|\mathbf{v}_i - \mathbf{v}_i^t\|^2 + 2\mathbf{g}^T\mathbf{x}_i\right) + \frac{\xi}{2}\overline{v_i}\|\mathbf{v}_i - \mathbf{v}_{air}\|^2,$$
$$\mathcal{E}_{e,j} := \frac{1}{2}\rho I\overline{e_j}\|\omega_j - \omega_j^t\|^2,$$

with $\mathbf{v}_i^t$ (resp. ω_j^t) the begin-of-step linear (resp angular) velocity, and $\mathbf{v}_{air}$ the ambient air velocity.

Constraints. Rigid attachment constraints, which we express formally as the inclusion $(\mathbf{x}, \theta) \in \mathcal{A}$, are enforced by kinematically fixing the two vertices and the twist of the clamped edges. At the n contact points, we write the Signorini–Coulomb conditions [*e.g,* Jean 1999] with friction coefficient μ and contact normal $\mathbf{n}$ on the local relative displacement $\mathbf{u} \in R^{3\times n}$ and contact force $\mathbf{r} \in \mathbb{R}^{n\times 3}$ as the inclusion $(\mathbf{u}, \mathbf{r}) \in C_{\mu,\mathbf{n}}$. Given the thin nature of hair strands, we assume that contact points always lie exactly on the centerline, thus can be expressed as a combination of the vertex positions $\mathbf{x}$, in line with Assumption 1 from [Daviet 2020]. This prevents contact forces from inducing edge torques but leads to significant simplifications in the contact solve. Formally, there exists an affine relationship $\mathbf{u} = H\mathbf{x} + \mathbf{u}^{\text{kin}}$, with $\mathbf{u}^{\text{kin}}$ gathering kinematic forcing terms.

4.2 Time integration

We use backwards Euler integration, such that the (unknown) end-of-step positions, velocities and displacements over a timestep Δ_t are related as $\Delta\mathbf{x} = \mathbf{x}^{t+1} - \mathbf{x}^t = \Delta_t\mathbf{v}^{t+1}$, $\Delta\theta = \theta^{t+1} - \theta^t = \Delta_t\omega^{t+1}$. In the following we write equations in terms of positions and displacements only as they remain well-defined in the quasistatic setting.

Incremental problem. We express our incremental problem as [1]

$$\min_{(\mathbf{x},\theta)\in\mathcal{A},(H\mathbf{x}+\mathbf{u}^{\text{kin}},\mathbf{r})\in C_{\mu,\mathbf{n}}} \mathcal{E}_s(\mathbf{x}) + \mathcal{E}_b(\mathbf{x},\theta) + \mathcal{E}_e(\mathbf{x},\theta). \quad (1)$$

[1] As remarked by Daviet [2020], this is actually an abuse of notation, due to the non-convexity of the Coulomb law with $\mu > 0$. However, this notation yields intuition about the objective splitting reformulations, and as we validate in Section 5.1 we do end up solving the correct problem, though without theoretical convergence guarantees.

A classical way to tackle this problem [*e.g,* Daviet 2020; Jean 1999; Kaufman et al. 2014] is to perform Newton iterations to reduce Problem (1) to a series of linear systems with frictional contact constraints, then solve each instance with any of the many dedicated solvers. However, for DER the left-hand-side is a band matrix with 10 sub- and super-diagonals, often poorly conditioned, leading to inefficient GPU memory accesses and potentially requiring double-precision solves. Alternatively, Problem (1) can be dealt with using purely local solvers such as XPBD [Macklin et al. 2016]. Constraint-coloring techniques can lead to a highly-parallel implementation, but propagate strains to the direct neighbors only at each iteration. While competitive for short rods, this approach results in many iterations being necessary to achieve convergence for large number of elements. We are therefore looking for a compromise between purely global (like Newton) and purely local (like XPBD) methods, *i.e,* with a left-hand-side that can be efficiently solved on the GPU but that can still propagate strains globally. A tridiagonal left-hand-side would fit the bill, and is what we now aim to construct.

Objective splitting. Following Daviet [2020]; Narain et al. [2016], we apply ADMM to decompose Problem (1) into easier subproblems. However, as ADMM trades convergence speed for a simpler global system, we do not blindly give the ADMM treatment to every energy term, instead making sure to keep the introduction of auxiliary variables to the minimum necessary to achieve our desired system shape. We introduce two primal variables, $\mathbf{y}$ and $\mathbf{z}$, and two dual variables, λ_y and λ_z, to reformulate Problem (1) equivalently as the optimization of an Augmented Lagrangian,

$$\min_{\substack{(\mathbf{x},\theta)\in\mathcal{A},\\ \mathbf{y}\in\mathbb{R}^{6\times m},\\ \left(H\mathbf{z}+\mathbf{u}^{\text{kin}},\mathbf{r}\right)\in C_{\mu,\mathbf{n}}}} \max_{\substack{\lambda_y\in\mathbb{R}^{6\times m},\\ \lambda_x\in\mathbb{R}^{3\times m}}} \mathcal{L}(\mathbf{x},\theta,\mathbf{y},\mathbf{z},\lambda_y,\lambda_z), \quad (2)$$

$$\begin{aligned}\mathcal{L}(\mathbf{x},\theta,\mathbf{y},\mathbf{z},\lambda_y,\lambda_z) :=& \mathcal{E}_s(\mathbf{y}) + \mathcal{E}_b(\mathbf{y},\theta) + \mathcal{E}_e(\mathbf{x},\theta)\\ &+ \lambda_y^T W_y(\mathbf{y} - B\mathbf{x}) + \lambda_z^T W_z(\mathbf{z} - \mathbf{x})\\ &+ \frac{1}{2}\|\mathbf{y} - B\mathbf{x}\|_{W_y}^2 + \frac{1}{2}\|\mathbf{z} - \mathbf{x}\|_{W_z}^2.\end{aligned}$$

In Problem (2), the matrix B maps the positions $\mathbf{x} \in R^{3\times m}$ to pairs of consecutive edges $\mathbf{y} \in R^{6\times m}$; for any interior vertex i, $(B\mathbf{x})_i := (\mathbf{x}_i - \mathbf{x}_{i-1}; \mathbf{x}_{i+1} - \mathbf{x}_i)$. The linear relationship $B\mathbf{x} = \mathbf{y}$ is enforced through the Lagrange multiplier $\lambda_y \in \mathbb{R}^{6\times m}$ associated to a constraint with diagonal weights $W_y \in \text{diag}(\mathbb{R}^m)$. The elastic energies $\mathcal{E}_b$ and $\mathcal{E}_s$ are now expressed on the twists θ and those edge pairs $\mathbf{y}$. Like in [Daviet 2020], the collision constraint is now expressed on an auxiliary variable $\mathbf{z}$ which tracks the original positions $\mathbf{x}$ through a constraint with weights $W_z \in \text{diag}(\mathbb{R}^m)$ and Lagrange multiplier $\lambda_z \in \mathbb{R}^{3\times m}$. External energies $\mathcal{E}_e$ and attachment constraints remain expressed on the original position and twist variables $\mathbf{x}$ and θ .

Each ADMM iteration then optimizes over each variable in turn:

- Over $\mathbf{y}$: we evaluate the embarrassingly parallel local elasticity proximal operator, as described in Section 4.4.
- Over $\mathbf{z}$: we project the tentative positions onto the feasible set of collisions constraints. This roughly follows [Daviet 2020], we detail the relevant modifications in Section 4.5.

ALGORITHM 1: Outline of our timestep integration algorithm

Input: Begin-of-step step positions $(\mathbf{x}^t, \theta^t)$ and velocities $(\mathbf{v}, \omega)$
// Steps involving Δ_t are for the dynamic setting only
Perform proximity-based collision detection ;
Advect initial guess $\mathbf{x} \leftarrow \mathbf{x}^t + \Delta_t \mathbf{v}$, $\theta \leftarrow \theta^t + \Delta_t \omega$;
Update DER reference and material frames ;
for *each ADMM iteration* **do**
  (Optionally) Perform continuous-time collision detection ;
  Perform local elasticity solve to get updated $\mathbf{y}$ (Section 4.4) ;
  Perform feasible projection to get updated $\mathbf{z}$ (Section 4.5) ;
  Update Lagrange multipliers $\lambda_y \leftarrow \lambda_y + \mathbf{y} - B\mathbf{x}$,
  $\lambda_z \leftarrow \lambda_z + \mathbf{z} - \mathbf{x}$;
  Assemble and solve tridiagonal system $\hat{M}\Delta\mathbf{x} = \hat{\mathbf{f}}$ (Section 4.3) ;
  Update $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$, $\mathbf{v} \leftarrow \mathbf{v} + \Delta\mathbf{x}/\Delta_t$;
  Update DER reference and material frames ;
  Assemble and solve tridiagonal system $\frac{\partial^2 \mathcal{E}_b + \mathcal{E}_s}{\partial \theta^2} \Delta\theta = -\frac{\partial E_b + E_s}{\partial \theta}$;
  Update $\theta \leftarrow \theta + \Delta\theta$, $\omega \leftarrow \omega + \Delta\theta/\Delta_t$;
  Update DER reference and material frames ;
end
(Optionally) Geometric contact correction: Repeat continuous-collision detection and feasible projection on $\mathbf{x}$;

- Over $a\lambda_y$ and λ_z: we do an explicit gradient step in the current residual direction, *i.e*, $\lambda_y \longleftarrow= \lambda_y + W_y(\mathbf{y} - B\mathbf{x})$, $\lambda_z \leftarrow \lambda_z + W_z(\mathbf{z} - \mathbf{x})$.
- Over $\mathbf{x}$: we solve three independent SPD tridiagonal systems of size m, one for each euclidean axis. See Section 4.3.
- Over θ: we do a single Newton step, solving for $\frac{\partial^2 \mathcal{E}_b + \mathcal{E}_s}{\partial \theta^2} \Delta\theta = -\frac{\partial E_b + E_s}{\partial \theta}$. This yields another SPD tridiagonal system of size m, which we solve in a similar manner.

Each time we update $\mathbf{x}$ or θ, we also update the reference and material DER frames using time-parallel transport. The full timestep integration scheme is outlined in Algorithm 1.

4.3 Global tridiagonal solve

The global step consists in performing the minimization over $\mathbf{x}$ in Problem (2). Denoting by $\Pi_{\mathcal{A}}$ the linear orthogonal projection operator associated to the kinematic attachment constraint $\mathcal{A}$, and $\Pi_{\mathcal{A}^\perp}$ its orthogonal complement, we need to solve the system

$$\underbrace{\left[\Pi_{\mathcal{A}^\perp} M \Pi_{\mathcal{A}^\perp}^T + \Pi_{\mathcal{A}} \Pi_{\mathcal{A}}^T\right]}_{\hat{M}} \Delta\mathbf{x} = \underbrace{\Pi_{\mathcal{A}^\perp}^T \mathbf{f}}_{\hat{\mathbf{f}}}, \tag{3}$$

$$\text{with } M := \frac{\partial^2 \mathcal{E}_e}{\partial \mathbf{x}^2} + W_z + B^T W_y B,$$

$$\mathbf{f} := -\frac{\partial \mathcal{E}_e}{\partial d\mathbf{x}} + W_z(\lambda_z + \mathbf{z} - \mathbf{x}) + B^T W_y(\lambda_y + \mathbf{y} - \mathbf{x}).$$

Here $\frac{\partial^2 \mathcal{E}_e}{\partial \mathbf{x}^2} + W_z$ is a diagonal matrix with positive coefficients (strictly positive in the dynamic setting), and $B^T W_y B$ can be re-ordered into three independent tridiagonal matrices, one for each euclidean coordinate axis or $\mathbb{R}^3$, and is positive semi-definite with three null modes corresponding to constant translations in $\mathbb{R}^3$. In the dynamic setting or assuming at least one kinematically fixed vertex, Eq. (3) thus boils down to solving three separate SPD tridiagonal systems of dimension m, $\hat{M}_{|k}\Delta\mathbf{x}_{|k} = \hat{\mathbf{f}}_{|k}$ for $k = 1 \dots 3$.

Parallel tridiagonal solves. Tridiagonal systems can be solved sequentially in linear time with the well-known Thomas algorithm. While parallelization over independent rods would be possible, uncoalesced memory accesses lead to poor practical performance unless all rods have the same number of vertices so that strided storage can be used [Valero-Lara et al. 2018]. Instead, we use Parallel Cyclic Reduction [PCR; *e.g*, Zhang et al. 2010], which needs only $\log m_\infty$ sequential steps, with m_∞ the maximum number of vertices per rod. While boasting much nicer memory access patterns, for maximum performance PCR needs to fit its working buffers into the GPU shared memory. Concretely, for a per-thread-block shared memory limit of at least 64kb [2], this means that we can solve systems of size $m_\infty \leq 1024$ without inter-block synchronization, which was enough for all of our hair simulation tasks. In practice, we group rods together within CUDA thread blocks until this limit is reached. One remarkable feature is that PCR does not feature separate factorization and solve steps; as such, there is no benefit from pre-factoring the matrix. Conversely, this means that there is no downside for adaptively changing the ADMM constraint weights between iterations; we exploit this fact to dynamically update collision weights in Section 4.5.

4.4 Local elasticity solve

Combined bending and stretching. While a common strategy for PD- and ADMM-based solvers is to solve each energy term separately, here we choose to combine bending $\mathcal{E}_b$ and stretching $\mathcal{E}_s$ together. Consider a pair of edges $(\mathbf{e}_{i-1}, \mathbf{e}_i) \in R^6$; the stretching energy will oppose modes of deformation following the edge directions, while the bending energy will mostly resist deformation in planes orthogonal to the edges. Intuitively, the combined energy is therefore coercive over all of $\mathbb{R}^6$, leading to reasonably-conditioned local problems, while the split energies suffer from null modes. In the DER model from Section 4.1, while the bending energy $\mathcal{E}_{b,i}$ is defined per-vertex and thus maps one-to-one with edge pairs $\mathbf{y}_i$, the stretching energy is defined per-edge; to avoid double-counting, we halve its stiffness for both pairs containing the edge.

Proximal operator. Minimizing the Augmented Lagrangian in Problem (2) over $\mathbf{y}$ involves solving for each edge pair $\mathbf{y}_i$,

$$\min_{\mathbf{y}_i \in R^6} \mathcal{E}_{b,i}(\mathbf{y}_i) + \mathcal{E}_{s,i}(\mathbf{y}_i) + \frac{W_{y,i}}{2} \|(B\mathbf{x})_i - \lambda_{y,i} - \mathbf{y}_i\|^2. \tag{4}$$

We evaluate this operator approximately by doing a single step of Gauss–Newton. Thanks to their reasonable condition numbers, we can solve the resulting 6×6 linear systems efficiently using single-precision LDLT factorization without numerical pivoting.

Local frame. As remarked by Brown and Narain [2021], rotations can lead to slow ADMM convergence. Indeed, the operator $B^T W_y B$ contains null modes corresponding to global translations, but not to global rotations, as the global energy hessian $\frac{\partial^2 \mathcal{E}_s + \mathcal{E}_b}{\partial \mathbf{x}^2}$ would. To compensate, we can attempt to perform the local solve in a rotation-invariant frame, then inject directly the rotated quantities into the

[2] this condition is satisfied by all GPUs with CUDA Compute Capability ≥ 7.5

global solve. Formally, we replace the matrix B with BQ^t, with $Q^t = \mathrm{diag}(Q_i^t)$ and and $(Q_i^t)_{i=1..m}$ per-constraint 3×3 rotation matrices. We experimented with defined Q_i^t from the current average material frame at each vertex, however we found this frame to be too noisy in the presence of collisions. Instead, we settle with defining as Q_i^t on a per-rod basis as the current root attachment frame.

Dynamic damping. We can incorporate strain-rate proportional damping in this framework by simply adding two supplemental terms to the proximal operator minimization (4) [Brown et al. 2018],

$$\mathcal{D}_{s,i}(\mathbf{y}_i) := \frac{\tau A E_s}{4\Delta_t}(\overline{e_{i-1}}\left(\varepsilon(\mathbf{e}_{i-1}) - \varepsilon_{i-1}^t\right)^2 + \overline{e_i}\left(\varepsilon(\mathbf{e}_i) - \varepsilon_i^t\right)^2),$$
$$\mathcal{D}_{b,i}(\mathbf{y}_i) := \frac{\tau E_b}{2\Delta_t}\left(\kappa_i\left(\mathbf{e}_{i-1}, \mathbf{e}_i\right) - \kappa_i^t\right)^T \frac{K}{\overline{v_i}}\left(\kappa_i\left(\mathbf{e}_{i-1}, \mathbf{e}_i\right) - \kappa_i^t\right),$$

with τ the typical relaxation time and ε_i^t (resp. κ_i^t) the stretching (resp. bending) strain values at the beginning of the timestep.

Elasticity constraint weights. As discussed by Overby et al. [2017], a sensible choice for the constraint weights W_y is to pick them close to the effective stiffness of the constraint, *i.e*, have the proximal penalization term in Eq. (4) be of a magnitude similar to that of the elastic energy $\mathcal{E}_s + \mathcal{E}_b$. To this end we choose $W_{y,i} := w^0 \frac{E_b A R^2}{4\overline{v_i}^3}$, where w^0 is a constant factor that accounts for the anisotropy and nonlinearities of $E_s + E_b$. In practice we pick $w^0 = 25$.

4.5 Feasible contact projection

The local solve associated to the collision constraint, *i.e*, the feasible contact projection, follows the procedure from Daviet [2020] but with emphasis on parallelization. We need to solve the Discrete Frictional Contact Problem (5) for $\mathbf{z} \in \mathbb{R}^{3\times m}$ and $\mathbf{u}, \mathbf{r}$ in $\mathbb{R}^{3\times n}$,

$$\begin{cases} W_z\mathbf{z} = W_z(\mathbf{x} - \lambda_z) + H\mathbf{r}^T \\ \quad \mathbf{u} = H\mathbf{z} + \mathbf{u}^{\mathrm{kin}} \\ (\mathbf{u}, \mathbf{r}) \in C_{\mu,\mathbf{n}}, \end{cases} \tag{5}$$

where we recall that W_z is diagonal with positive coefficients, and H is such that for each contact c, $(Hz)_c := \sum_{i\in\mathcal{I}_c} h_{c,i}\mathbf{z}_i$ is a linear combination of some subset $\mathcal{I}_c$ of vertex positions $(\mathbf{z}_i)$.

Collision detection. We perform proximity-based collision detection at the beginning of each timestep and optionally continuous-time at a subset of the ADMM iterations. We use a shallow acceleration structure consisting of one sparse hash grid will cell sizes equal to the longest rod edge, with a single uniform dense subdivision level inside each sparse cell. Broad-phase collision culling is performed using 18-dops [Klosowski et al. 1998], yielding a good compromise between bounding tightness and memory bandwidth usage. Note that edge dops are symmetric about each axis and can be stored using only 12 floats. To reduce the number of redundant contacts and simplify preallocating buffers, in the proximity phase we discard any contact which is not among the 6 closest ones for any of the colliding edges. For continuous-time collision detection we keep only the two earliest impact times for each edge, estimated using ACCD [Li et al. 2021]. Collision against kinematic meshes are computed by bisecting signed-distance functions [Macklin et al. 2020], either from a voxelized representation for static meshes [Museth 2021] or direct closest-point queries for animated meshes.

Hybrid contact solve. Daviet [2020] resolves contacts in a Gauss–Seidel fashion, using contact-coloring heuristics to find subsets of contacts that can be solved in parallel. However, Gauss–Seidel will always require a number of sequential steps greater than the maximum number of contacts per vertex, $\max_{i\le m} n_{|i}$, which can be in the hundreds in our case (*e.g*, for a stray hair orthogonal to a dense wisp). This is adequate for latency-optimized processors with relatively low core count, but results in sub-optimal and unbalanced workloads on GPUs. Instead we propose an hybrid strategy to maximize throughput: we heuristically split contacts into a small number (8 in our implementation) of batches, processed one after the other in a Gauss–Seidel fashion; then solve contacts within each batch in a fully parallel Jacobi fashion. To ensure determinism, we proceed for each batch in two steps:

(1) For each contact c in parallel, compute $\mathbf{u}_c^* := (H\mathbf{z})_c + \mathbf{u}_c^{\mathrm{kin}}$, solve for $\Delta\mathbf{u}_c$ such that $(\Delta\mathbf{u}_c + \mathbf{u}_c^*, \Delta\mathbf{u}_c + \tilde{\mathbf{r}_c}) \in C_{\mu_c,\mathbf{n}_c}$ [see Daviet 2020, Sec. 5.2], then update $\tilde{\mathbf{r}_c} \leftarrow \tilde{\mathbf{r}_c} + \Delta\mathbf{u}_c$. Here $\tilde{\mathbf{r}^c}$ keeps track of the total applied correction by the contact c and is initialized to zero when c is first solved. [3].
(2) Update vertex positions in parallel as $\mathbf{z} \leftarrow \mathbf{z} + W_z^{-1}H^TD^{-1}\Delta\mathbf{u}$ with D the diagonal of the Delassus operator,

$$D := \mathrm{diag}\left(\sum_{i\in\mathcal{I}_c} h_{c,i}^2/W_{z,i}\right)_{c=1\ldots n.} \tag{6}$$

Note that the sparse matrix $W_z^{-1}H^TD^{-1}$ is constant and can be preassembled after the collision detection step.

Now, Jacobi has weaker convergence guarantees than Gauss–Seidel. Imagine a single contact duplicated as c_1 and c_2; solving them in parallel with Jacobi will lead to an update $\Delta\mathbf{u}_{c_1} + \Delta\mathbf{u}_{c_2}$ equal to twice the necessary correction, overshooting the expected result and failing to decrease the residual. One strategy to ensure convergence is under-relaxation, dividing the update from contact c $\Delta\mathbf{u}_c$ by the maximum number of contacts across all nodes involved in c, $\max_{i\in\mathcal{I}_c} n_{|i}$. This does address robustness issues, but is over-conservative and leads to a sub-optimal rate of convergence. Instead, we borrow ideas from Li et al. [2018]; Tonge et al. [2012] and conceptually split each vertex i into $n_{|i}$ virtual vertices, each being assigned $1/n_{|i}$ of the original vertex weight $W_{z,i}$, then update $\mathbf{z}_i$ with the average correction from all duplicates. Each contact being now effectively independent, we get back the theoretical robustness guarantees of Gauss–Seidel. Concretely, we simply need to replace the Delassus operator diagonal D from Eq. (6) with $\hat{D}$,

$$\hat{D} := \mathrm{diag}\left(\sum_{i\in\mathcal{I}_c} n_{|i}h_{c,i}^2/W_{z,i}\right)_{c=1\ldots n}.$$

Not that our algorithm using the modified $\hat{D}$ still converges to the solution of Problem 5.

Collision constraint weights. Daviet [2020] scales the collision constraint weight with inertia, *i.e*, $W_{z,i} \sim m_i/\Delta_t^2$. However in the quasistatic setting Δ_t is not finite; we replace it with $\widetilde{\Delta_t}$, a typical

[3]The accumulated displacement $\tilde{\mathbf{r}}$ is related to the contact force $\mathbf{r}$ as $\tilde{\mathbf{r}} = D\mathbf{r}$, with D the Delassus operator diagonal from Eq. (6).

time that we compute from dimensional analysis on the external acceleration $\mathbf{g}$, bending stiffness k_b and typical edge length $\tilde{L}$ as

$$\widetilde{\Delta_t} := \min\left(\Delta_t, \widetilde{\Delta_t}^g, \widetilde{\Delta_t}^b\right), \quad \widetilde{\Delta_t}^g := \sqrt{\tilde{L}/\|\mathbf{g}\|},$$
$$\widetilde{\Delta_t}^b := \sqrt{\tilde{L}/\left(\frac{k_b}{\tilde{L}}\right)} = \sqrt{\rho\tilde{L}^2/\left(\frac{E_b R^2}{4\tilde{L}^2}\right)} = 2\frac{\tilde{L}^2}{R}\sqrt{\frac{\rho}{E_b}}.$$

Vertices not involved in any collision are assigned a zero weight.

5 RESULTS

All simulations were performed on a workstation equipped with an Intel Core i9-10980XE CPU and two NVIDIA GeForce 3080 Ti GPUs with 12 GB of memory each (unless otherwise mentioned, only one GPU was used for the simulations presented below). Renderings were done using NVIDIA Omniverse.

5.1 Validation

Romero et al. [2021] presented multiple experiments designed to test the consistency of predictions coming from numerical simulators, analytic derivations and physical experiments. Three of these experiments — *cantilever*, *bend–twist* and *stick-slip* — are relevant for elastic rods, and we therefore verify that our simulator is in agreement with their results. In order to benefit from the massive parallelism of our solver, for each experiment we solve for all configurations at once. Results are shown in Fig. 2 (see also our supplemental video) and discussed below.

Cantilever. The cantilever experiment computes the equilibrium of naturally straight rods of varying length under gravity, and compares the resulting aspect-ratio H/W to theoretical predictions. We ran our simulator in a quasistatic setting for a total of 5,000 iterations, which completed in less than one second for the 100 configurations. We noticed that our original results (dashed blue line in Fig. 2 left) slightly underestimated the theoretical aspect ratio. We explain this discrepancy from the fact that the bending energy is integrated only over the Voronoi length $\overline{v_i}$ associated to the interior vertices i; the first half of the root edge and the last half of the tip edge are thus excluded from bending computations. Changing the definition of the Voronoi length associated to the second and penultimate vertices to include the full length of boundary edges, $\overline{v_1}^* := \overline{e_0} + \overline{e_1}/2$ and $\overline{v_{m-2}}^* := \overline{e_{m-1}} + \overline{e_{m-2}}/2$, we get a much better match to the analytical results (red crossed line). Note that this effect is inversely proportional to the resolution, so the simulator should still converge to the ground truth without Voronoi length rescaling.

Bend–twist. The bend–twist test consists in setting up rods with various length and natural curvatures, and determining whether their equilibrium under gravity is planar — which we do in practice by performing a PCA on the equilibrium shape and comparing the minimum and average eigenvalues. Romero et al. [2021] mentions that long, hook-shaped planar configurations can take a long time to converge; this is also true for our simulator, and we observed that we reached equilibrium faster in a damped dynamic setting than in pure quasistatic setting. This could likely be remedied to by introducing numerical momentum in the quasistatic ADMM optimization, but we leave this investigation for future work. Letting the simulation run for 2500 timesteps — the gravity being tilted for the first 100 timesteps to induce a transverse perturbation — and using 100 ADMM iterations per timestep — a few seconds of total runtime — we retrieve a good agreement between our results and the analytical curve.

Table 1. Performance statistics

Scene	#vertices	#contacts	t_{frame} (s)[a]	Peak GB
Hairball 16k	480k	1.41M	0.18	1.1
Hairball 16k, CT	480k	2.10M	0.29	1.3
Hairball 128k	3.8M	27M	5.6	9.8
Hairball 128k, CT	3.8M	34M	6.78	2×9.4[b]
Long 10k, CT	339k	1.7M	0.28	0.9
Long 47k, CT	1.4M	10M	3.5	5.4
Curly 24k, CT	617k	1.5k	0.25	1.4

[a] t_{frame} indicates the average computation time for a full 24fps frame, using 4 timesteps per frame and 10 ADMM iterations per timestep for all the examples in this table.
[b] The scene "128k, CT" was run on 2 GPUs

Stick–slip. For the stick–slip test we progressively push a vertically clamped, naturally straight rod against a fixed plane, and record the vertical displacement at which the buckling of the rod leads it to overcome frictional forces. This critical displacement increases with the friction coefficient, until a critical value $\mu_c \sim 0.36$ is reached above which sliding is no longer possible [Romero et al. 2021]. The results from our simulator, using a total of 800 substeps to progressively attain the maximum displacement of $0.6L$, are once again in good agreement with the theory.

5.2 Performance

While the previous section focused on eventual convergence of the solver, we now try to evaluate its computational performance.

Hairy balls. We first reproduce the "hairy ball" experiment from Daviet [2020]; Kaufman et al. [2014]; see Fig. 3 and the accompanying video. While our timings cannot be directly compared to that of previous works, as the solvers were run on different generations of hardware, and with small variations in the strands geometry and physical properties, we believe this test can still yield useful insights about the scalability of the different techniques.

Following Daviet [2020], we implant strands of radius $R = 0.037mm$, length $L = 30cm$, and friction coefficient $\mu = 0.2$ over the top half of a ball with radius 9cm, and subject them to the sequence of rotations described by Kaufman et al. [2014] at 24 frames per second and 4 substeps per frame. We add slight natural curvature and twist defined by an helical radius of 0.5cm and a pitch of 5cm. Like all other simulations in this work, we use $\rho = 1300\text{kg.m}^{-3}$, $\|\mathbf{g}\| = 9.81\ \text{m.s}^{-1}$, $E_b = 3\text{GPa}$ and $\nu = 0.45$. We use 10 ADMM iterations per timestep, and run 2 Gauss–Seidel/Jacobi feasible projection iterations at each ADMM iteration, plus 10 more for the "geometric correction" step at the end of the timestep.

We ran simulations for balls with 16k and 128k strands, each with and without continuous-time collisions. In the former case, detection is performed 3 times per timestep at regular intervals. Due to exceeding the memory budget of a single GPU, the "Hairball

128k, CT" scene was run over 2 GPUs, using the same mass-splitting technique described in Section 4.5 to handle contacts involving vertices residing on different GPUs. Table 1 summarize out timings; the 16k configuration can run at interactive frame rates, while the full-resolution 128k balls require about 40 minutes to finish — to be compared to half-a-week with the CPU-based implementation of Daviet [2020], and untractable memory requirements for the solver from [Kaufman et al. 2014].

Realistic grooms. To verify the performance of our solver on more realistic assets, we also simulate two grooms consisting of 9,600 (resp. 47,100) long, mostly straight strands, and a third one with 23,700 highly curly strands (Fig. 4). Physical and numerical parameters are kept identical to that of the hairy ball simulations, and the grooms are subjected to a series fast head turns. Timings are reported in Table 1 and broken down in Fig. 4, bottom.

5.3 Applications

The computational efficiency of our simulator enables applications beyond classical offline simulations; we discuss some of them below.

Physics-based grooming. We imagine a primitive physics-based editing tool that can be used to make adjustments to an existing groom, while respecting elasticity and self-collision constraints, ensuring preservation of the layering and volume of the hair, and providing an accurate preview of what the groom will look at simulation time. For demonstration purposes our tool allows: uniformly scaling the hair length and/or curvature; trimming the rods along a cutting plane; and direct manipulation of strands within a selection radius through spring-like forces.

The usability of such a tool highly depends on its response time — which also includes rendering, further limiting the simulation budget. On the other hand, since this application is quasistatic, we do not need to wait for the optimization to converge before displaying intermediate results, and can therefore afford a low number of ADMM iterations per rendered frame. Furthermore, as the relative strand motion between rendered frames is small, we can afford to rely solely on proximity-based collisions. The resulting computational budget being still too tight to allow simulation of every single groom strand, we take inspiration from geometric grooming tools and resort to a guide-based approach where only 10% of the strands are actually simulated, and the rest is deformed according to the guides motion. Here we use linear–blend–skinning, but more advanced real-time deformation techniques, such as the neural interpolation from Lyu et al. [2022], could be employed.

This framework allows us to edit a groom with about 86,000 rendered curves (8600 curves being simulated, amounting to 120k vertices; see also Fig. 1 and the full editing session in the accompanying video) with real-time feedback. Each simulation frame runs 10 ADMM iterations, for an average wall time of 17ms.

Procedural edits. We also attempted to use this tooling in a procedural manner to efficiently explore creating variations from a single base groom. We can provide the simulator with a series of basic actions (apply animated external forces, growing/trimming hair, …) that can be combined in many different ways to quickly generate a large amount of candidate variants, some of which are presented on Figure 5. While most generated variants not may not be of interest, the fast iteration time of the overall process allows to quickly identify and focus over the ones that are.

6 DISCUSSION

6.1 Limitations and future work

While being significantly cheaper than Newton per-iteration [4] the convergence of ADMM is not as well understood. We have proposed heuristics to scale the constraint weights with the simulated scene, and did not need to tune hyperparameters for the examples presented in this article; however the performance of our ADMM optimizer on physical systems other than typical human hairs remains unclear. Overall the solver will perform best for homogeneous and isotropic materials, and grooms with high variance in edge lengths or elastic moduli, or very different stretching and bending stiffnesses, will negatively impact convergence.

It is also unclear how the validation of our solver on the model problems from Romero et al. [2021] translates to large scenes for which we cannot afford to run as many iterations of the ADMM optimizer. Imagining validation experiments involving larger contacting rod assemblies would be a first step toward understanding this behavior. Our complementarity-based contact solver does not feature the same intersection-free guarantees as recent barrier methods derived from Li et al. [2020]; while missing a few collisions for hair is much less dramatic than for cloth, it can still result in undesired entanglement as evidenced for the proximity-only "Hairball 128k" scene in Fig. 3. Combining our highly-scalable solver with timestep clamping ideas from [Li et al. 2020] is a promising direction.

Multiple phenomena are still unaccounted for by our solver, such as the effect of hair products, interactions with the surrounding air, or two-way coupling with soft bodies. Simulating those in an efficient manner within our framework remains to be investigated.

Finally, for physics-based grooming to become viable, several UX challenges remain to be solved. Haptic feedback and smart selection tools allowing to manipulate, comb and trim sections of the hair at desired angles could help make the experience closer to that of a real-life barbershop, but would still require highly trained artists.

6.2 Conclusion

Through a carefully chosen splitting of the timestep incremental problem objective, we have devised a simulation algorithm for Kirchhoff rods under dry frictional contacts that maps efficiently to the massively parallel architectures of modern GPUs, and verified that it is able to reproduce analytical results on multiple experiments. Our simulator brings a leap in performance compared to state-of-the-art CPU-based global solvers, reducing multiple-day computation times to merely hours, and finally allowing to envision interactive physics–based hair simulation and grooming at high strand counts.

ACKNOWLEDGMENTS

The author would like to thank the anonymous referees, the High–Fidelity Physics group led by Ken Museth at NVIDIA, as well as the Omniverse Digital Human and RTX teams for their support.

[4] Ignoring collisions, on our "realistic grooms" we observe one ADMM iteration to be $\sim 30\times$ cheaper than a Newton iteration solved using the cuSPARSE sbric02 routine.

REFERENCES

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete Viscous Threads. *ACM Trans. Graph.* 29, 4, Article 116 (jul 2010), 10 pages.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete Elastic Rods. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–12.

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.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (jul 2014), 11 pages.

George E. Brown and Rahul Narain. 2021. WRAPD: Weighted Rotation-Aware ADMM for Parameterization and Deformation. *ACM Trans. Graph.* 40, 4, Article 82 (jul 2021), 14 pages.

George E. Brown, Matthew Overby, Zahra Forootaninia, and Rahul Narain. 2018. Accurate Dissipative Forces in Optimization Integrators. *ACM Trans. Graph.* 37, 6, Article 282 (dec 2018), 14 pages.

Romain Casati and Florence Bertails-Descoubes. 2013. Super Space Clothoids. *ACM Trans. Graph.* 32, 4, Article 48 (jul 2013), 12 pages.

Byoungwon Choe, Min Gyu Choi, and Hyeongseok Ko. 2005. Simulating complex hair with robust collision handling. In *Symposium on Computer Animation*.

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, Florence Bertails-Descoubes, and Laurence Boissieux. 2011. A Hybrid Iterative Solver for Robustly Capturing Coulomb Friction in Hair Dynamics. *ACM Trans. Graph.* 30, 6 (dec 2011), 1–12.

Alexandre Derouet-Jourdan, Florence Bertails-Descoubes, Gilles Daviet, and Joëlle Thollot. 2013. Inverse Dynamic Hair Modeling with Frictional Contact. *ACM Trans. Graph.* 32, 6, Article 159 (nov 2013), 10 pages.

Crispin Deul, Tassilo Kugelstadt, Marcel Weiler, and Jan Bender. 2018. Direct Position-Based Solver for Stiff Rods. *Computer Graphics Forum* 37 (2018).

Sunil Hadap and Nadia Magnenat-Thalmann. 2001. Modeling Dynamic Hair as a Continuum. *Computer Graphics Forum* 20 (2001).

Jerry Hsu, Nghia Truong, Cem Yuksel, and Kui Wu. 2022. A General Two-Stage Initialization for Sag-Free Deformable Simulations. *ACM Trans. Graph.* 41, 4, Article 64 (jul 2022), 13 pages.

Hayley Iben, Jacob Brooks, and Christopher Bolwyn. 2019. Holding the Shape in Hair Simulation. In *ACM SIGGRAPH 2019 Talks* (Los Angeles, California) *(SIGGRAPH '19)*. Association for Computing Machinery, New York, NY, USA, Article 59, 2 pages.

M. Jean. 1999. The non-smooth contact dynamics method. *Computer Methods in Applied Mechanics and Engineering* 177, 3 (1999), 235–257.

Danny M. Kaufman, Rasmus Tamstorf, Breannan Smith, Jean-Marie Aubry, and Eitan Grinspun. 2014. Adaptive Nonlinearity for Collisions in Complex Rod Assemblies. *ACM Trans. Graph.* 33, 4, Article 123 (jul 2014), 12 pages.

J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. 1998. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics* 4, 1 (1998), 21–36.

Petr Kmoch, Ugo Bonanni, and Josef Pelikán. 2010. Towards a GPU-Only Rod-Based Hair Animation System. In *ACM SIGGRAPH ASIA 2010 Posters* (Seoul, Republic of Korea) *(SA '10)*. Association for Computing Machinery, New York, NY, USA, Article 7, 1 pages.

K. Korner, B. Audoly, and K. Bhattacharya. 2021. Simple deformation measures for discrete elastic rods and ribbons. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 477, 2256 (2021), 20210561.

Tassilo Kugelstadt and Elmar Schömer. 2016. Position and Orientation Based Cosserat Rods. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, Ladislav Kavan and Chris Wojtan (Eds.). The Eurographics Association.

Lei Lan, Guanqun Ma, Yin Yang, Changxi Zheng, Minchen Li, and Chenfanfu Jiang. 2022. Penetration-Free Projective Dynamics on the GPU. *ACM Trans. Graph.* 41, 4, Article 69 (jul 2022), 16 pages.

Steve Lesser, Alexey Stomakhin, Gilles Daviet, Joel Wretborn, John Edholm, Noh-Hoon Lee, Eston Schweickart, Xiao Zhai, Sean Flynn, and Andrew Moffat. 2022. Loki: A Unified Multiphysics Simulation Framework for Production. *ACM Trans. Graph.* 41, 4, Article 50 (jul 2022), 20 pages.

Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E. Brown, and Laurence Boissieux. 2018. An Implicit Frictional Contact Solver for Adaptive Cloth Simulation. *ACM Trans. Graph.* 37, 4, Article 52 (jul 2018), 15 pages.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection- and Inversion-free Large Deformation Dynamics. *ACM Trans. Graph. (SIGGRAPH)* 39, 4, Article 49 (2020).

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Trans. Graph.* 40, 4, Article 170 (jul 2021), 24 pages.

Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective Dynamics with Dry Frictional Contact. *ACM Trans. Graph.* 39, 4, Article 57 (aug 2020), 8 pages.

Q. Lyu, M. Chai, X. Chen, and K. Zhou. 2022. Real-Time Hair Simulation With Neural Interpolation. *IEEE Transactions on Visualization and Computer Graphics* 28, 04 (apr 2022), 1894–1905.

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Zach Corse. 2020. Local Optimization for Robust Signed Distance Field Collision. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 1, Article 8 (may 2020), 17 pages.

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)*. Association for Computing Machinery, New York, NY, USA, 49–54.

Miles Macklin, Kier Storey, Michelle Lu, Pierre Terdiman, Nuttapong Chentanez, Stefan Jeschke, and Matthias Müller. 2019. Small Steps in Physics Simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '19)*. Association for Computing Machinery, New York, NY, USA, Article 2, 7 pages.

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.

Matthias Müller, Nuttapong Chentanez, Miles Macklin, and Stefan Jeschke. 2017. Long Range Constraints for Rigid Body Simulations. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '17)*. Association for Computing Machinery, New York, NY, USA, Article 14, 10 pages.

Matthias Müller, Tae-Yong Kim, and Nuttapong Chentanez. 2012. Fast Simulation of Inextensible Hair and Fur. In *Workshop on Virtual Reality Interactions and Physical Simulations*.

Ken Museth. 2021. NanoVDB: A GPU-Friendly and Portable VDB Data Structure For Real-Time Rendering And Simulation. In *ACM SIGGRAPH 2021 Talks* (Virtual Event, USA) *(SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 1, 2 pages.

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Zurich, Switzerland) *(SCA '16)*. Eurographics Association, Goslar, DEU, 21–28.

Masaki Oshita. 2007. Real-time hair simulation on GPU with a dynamic wisp model. *Computer Animation and Virtual Worlds* 18, 4-5 (2007), 583–593.

Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM ⊇ Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (Oct 2017), 2222–2234.

Dinesh K. Pai. 2002. STRANDS: Interactive Simulation of Thin Solids using Cosserat Models. *Computer Graphics Forum* 21, 3 (2002), 347–352.

Lena Petrovic, Marc Henne, and John Anderson. 2005. *Volumetric Methods for Simulation and Rendering of Hair.* Technical Report. Pixar Animation Studios.

Victor Romero, Mickaël Ly, Abdullah Haroon Rasheed, Raphaël Charrondière, Arnaud Lazarus, Sébastien Neukirch, and Florence Bertails-Descoubes. 2021. Physical Validation of Simulators in Computer Graphics: A New Framework Dedicated to Slender Elastic Structures and Frictional Contact. *ACM Trans. Graph.* 40, 4, Article 66, 19 pages.

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

Carlota Soler, Tobias Martin, and Olga Sorkine-Hornung. 2018. Cosserat Rods with Projective Dynamics. *Computer Graphics Forum* 37, 8 (2018), 137–147.

Jonas Spillmann and Matthias Teschner. 2007. CoRdE: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Symposium on Computer Animation*.

Richard Tonge, Feodor Benevolenski, and Andrey Voroshilov. 2012. Mass Splitting for Jitter-Free Parallel Rigid Body Simulation. *ACM Trans. Graph.* 31, 4, Article 105 (jul 2012), 8 pages.

Pedro Valero-Lara, Ivan Martínez-Pérez, Raül Sirvent, Xavier Martorell, and Antonio J. Peña. 2018. cuThomasBatch and cuThomasVBatch, CUDA Routines to compute batch of tridiagonal systems on NVIDIA GPUs. *Concurrency and Computation: Practice and Experience* 30, 24 (2018), e4909. e4909 cpe.4909.

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.

Yao Zhang, Jonathan Cohen, and John D. Owens. 2010. Fast Tridiagonal Solvers on the GPU. In *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Bangalore, India) *(PPoPP '10)*. Association for Computing Machinery, New York, NY, USA, 127–136.

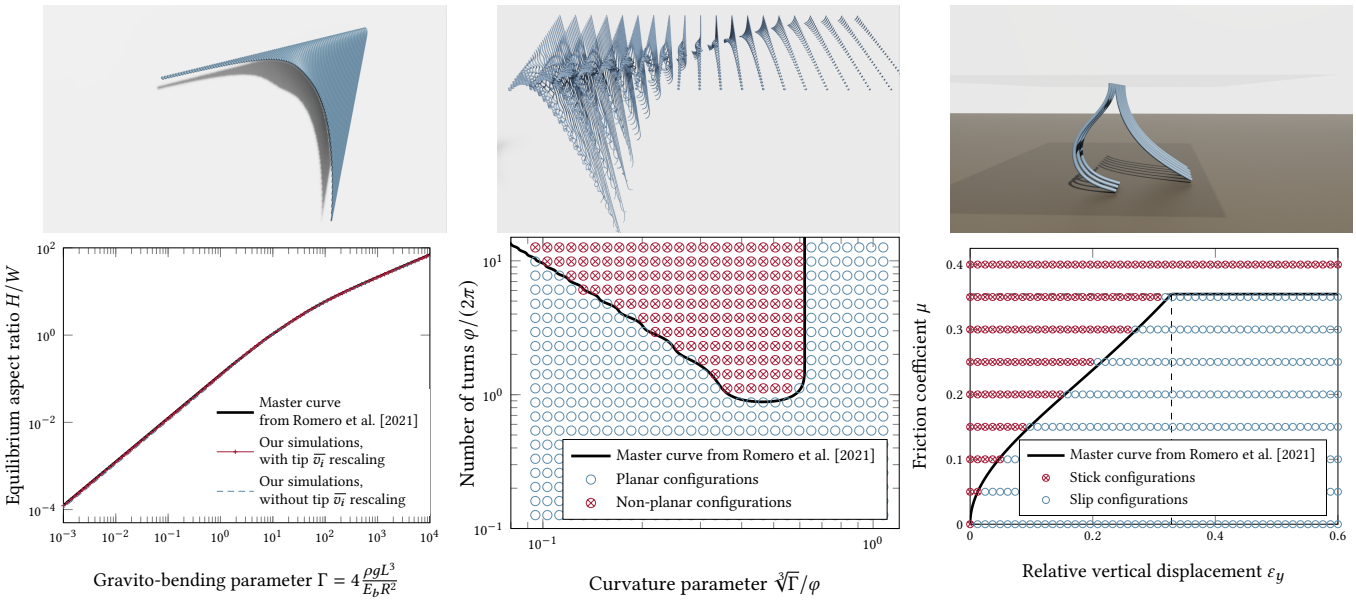

Fig. 2. Results for the numerical verification of our simulator on the cantilever (left), bend–twist (middle) and stick–slip (right) experiments from Romero et al. [2021]. Physical parameters in these tests are representative of human hair: R = 0.05mm, ρ = 1300 kg.m^{-3}, $\|\mathbf{g}\|$ = 9.81 m.s^{-1}, E_b = 3GPa, ν = 0.45. All results are shown for a resolution of 2.5mm per edge, with a minimum of 16 edges per rod. For the cantilever test, we varied the length from 0.5cm to 128cm to span 7 orders of magnitude of Γ. For the bend-twist test, we varied the natural curvature from 0.175 to 2 cm^{-1} and φ from 0.25π to 25π, leading to curve lengths ranging from 0.4 to 444cm. For the slip-stick experiment, we used rods of length 10cm.

(a) $t = 0.83s$ (b) $t = 5.41s$ (c) $t = 6.21s$ (d) $t = 9.79s$ (e) $t = 10.41s$ (f) $t = 15s$ (g) $t = 15s$, close-up

Fig. 3. Captures from our hair ball simulations, from top to bottom: 16k, no continuous-time; 16k, continuous-time; 128k, no continuous-time; 128k, continuous-time. Skipping continuous-time collision detection leads to a more clumped final state, as strands that get entangled due to missed collisions during the high-speed impact phases maintain proximity and cannot separate again at the later stages.

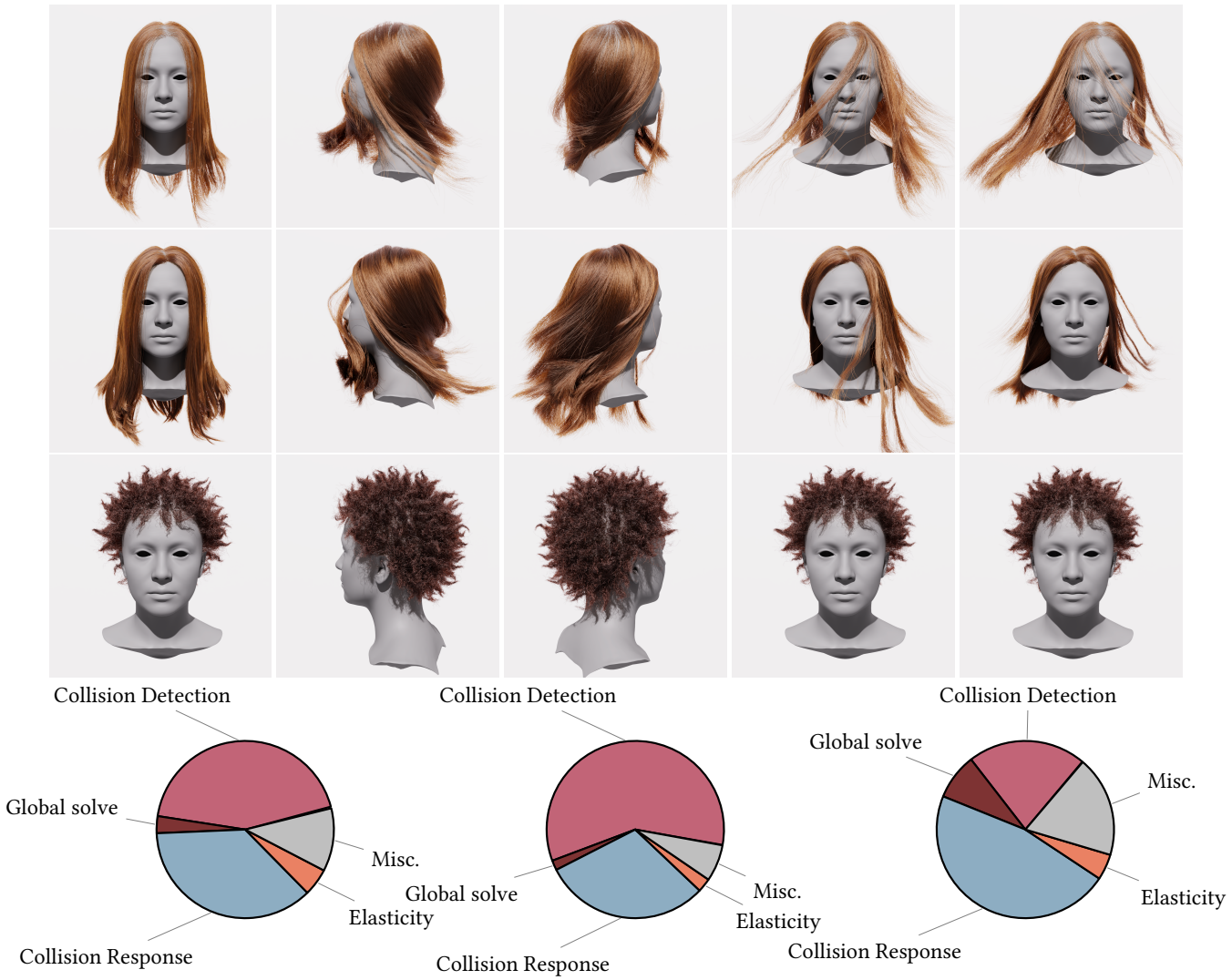

Fig. 4. Captures from the "Long 10k", "Long 47k" and "Curly 24k" simulations, and repartition of computation time for the first 20 frames of each simulation.

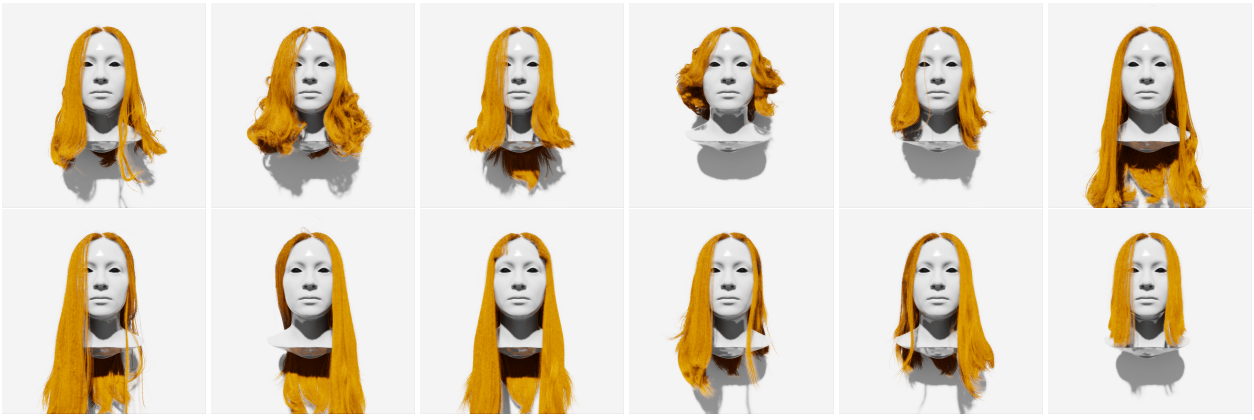

Fig. 5. Procedurally generating variants from a single input groom ("Long") by growing, curling, trimming rods and animating external acceleration.