Pacific Graphics 2020
E. Eisemann, A. Jacobson, and F.-L Zhang
(Guest Editors)

Volume 39 (2020), Number 7

Cosserat Rod with *rh*-Adaptive Discretization

Jiahao Wen[1], Jiong Chen[1], Nobuyuki Umetani[2], Hujun Bao[1], Jin Huang[1†]

[1]State Key Lab of CAD&CG, Zhejiang University
[2]The University of Tokyo

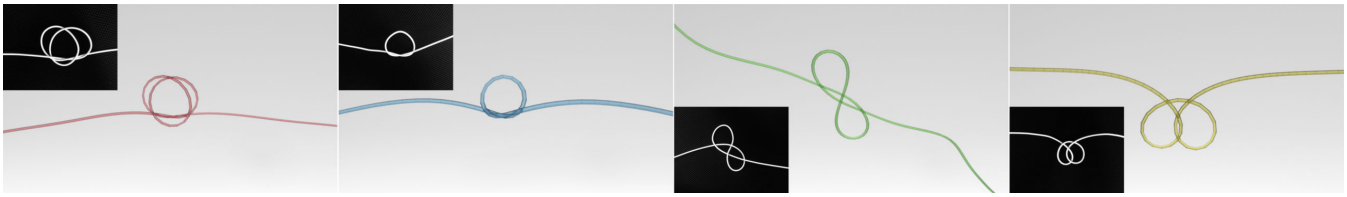


Figure 1: ***Knots in virtuality and reality****. Our method can effectively adapt the discretization to the geometry of deformed Cosserat rods as well as contact. As a result, it enables to simulate fascinating behaviors such as the dynamical evolution of shrinking knot loops, offering a faithful virtual reproduction of real-world rod dynamics.*

Abstract

Rod-like one-dimensional elastic objects often exhibit complex behaviors which pose great challenges to the discretization method for pursuing a faithful simulation. By only moving a small portion of material points, the Eulerian-on-Lagrangian (EoL) method already shows great adaptivity to handle sharp contact, but it is still far from enough to reproduce rich and complex geometry details arising in simulations. In this paper, we extend the discrete configuration space by unifying all Lagrangian and EoL nodes in representation for even more adaptivity with every sample being assigned with a dynamic material coordinate. However, this great extension will immediately bring in much more redundancy in the dynamic system. Therefore, we propose additional energy to control the spatial distribution of all material points, seeking to equally space them with respect to a curvature-based density field as a monitor. This flexible approach can effectively constrain the motion of material points to resolve numerical degeneracy, while simultaneously enables them to notably slide inside the parametric domain to account for the shape parameterization. Besides, to accurately respond to sharp contact, our method can also insert or remove nodes online and adjust the energy stiffness to suppress possible jittering artifacts that could be excited in a stiff system. As a result of this hybrid rh-adaption, our proposed method is capable of reproducing many realistic rod dynamics, such as excessive bending, twisting and knotting while only using a limited number of elements.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

Simulating ropes, cables, and other strand-like physical curves has always been an active research area in computer graphics. Due to its one dimensional intrinsic structure, such objects can have very delicate and complicated behaviors like buckling and knotting. Using a constant number of static nodes for discretization has an obvious limitation in striking a good balance between accuracy and efficiency. Since critical geometric features are generally unpredictable in advance, to accurately capture them, one always has to blindly use a large number of sample points for discretization. However, such a naive strategy will inevitably lead to huge computational cost, while using coarse discretization instead is generally not expressive enough to capture the rich geometric details appearing in simulation.

As an effective solution to make trade-offs between accuracy and computational cost, adaptive methods attempt to resolve the dilemma by seeking for smart strategies for resampling nodes. Among all possible choices, adaptively inserting and removing nodes is a quite common strategy which has been explored for

† Corresponding author: hj@cad.zju.edu.cn

years, even on the Cosserat model [ST08], which nicely captures some complex behaviors. Such a technique is well known as the *h*-adaptive FEM which adjusts spacings of elements if necessary by dynamically changing the sampling density. Another strategy is the *r*-adaptive method, sometimes also called as the moving mesh method [BHR09, HR10]. In graphics community, the recently popularized Eulerian-on-Lagrangian (EoL for short) method [FLLP13] shares similar insights with the *r*-adaptive method. The key idea is to move the existing nodes to proper places to account for violent dynamics without having additional ones. Combined with node inserting and removing strategy, the EoL method is capable of generating smooth dynamics without spurious jittering even with sharp contact, whose effectiveness has been well demonstrated on mass-spring model [SJLP11], cloth [WPLS18] and yarn-level model [CLMMO14]. It is intuitive to directly apply the methods by [SJLP11] to Cosserat model for a more realistic simulation, but we notice three limitations therein.

Firstly, the existing EoL methods are mostly contact-oriented. These methods generally put a node on the contact point and assign the added node with a changeable material (Eulerian) coordinate. These inserted nodes are marked as EoL nodes. Under the joint effect of collision response constraints and elastic forces, the Eulerian coordinate can slide inside the parametric domain to address possible locking issues. For the remaining nodes, they are just treated as pure Lagrangian whose material coordinate is always fixed, losing the chance to get adapted in regions far from the contact area. Thus, instead of treating EoL and Lagrangian nodes separately, we propose to unify them in representation by assigning *every* point with a time-variant material coordinate. In this way, all material points can be moved quite flexibly to adapt to rod shapes as well as contact for an accurate simulation. In principle, our approach better reveals the *r*-adaptive nature of the EoL technique and greatly extends the configuration space to discretize governing equations.

Secondly, when incorporating Eulerian coordinates as extra DoFs, the equation of motion will become singular because Eulerian and Lagrangian coordinates can be redundant in representing physical states. To remove this degeneracy, one need to constrain the motion of material points, and an important question to answer is how these points should be moved. In previous works [WPLS18], the Eulerian coordinate starts to slide if the Lagrangian ones are locked due to over-constraining, for instance, to respond to contact with sharp edges. Otherwise, they are restricted to stay as static as possible. However, the Cosserat rod can have much more complicated shapes due to the couplings of twist and bend, so only sliding material points near contact regions is quite insufficient. Ideally, the whole reference mesh should be more relaxed to move in a wider parametric range. Inspired by the moving mesh method, our approach requires that the spatial distribution of material points should be equally spaced with respect to curvature distribution, thus being able to reproduce complicated behaviors effectively.

Finally, to accurately handle contacts, improve asymptotic approximation, and avoid ill-conditioned system caused by tiny elements, our method also allows to dynamically insert or remove nodes as previous works did. However, every change to the discretization takes the risk of spurious jittering. An important criterion is to preserve shape and energy [ST08]. But geometric operations like node collapsing make it hard to achieve cheaply and could introduce drastically large fictitious ghost forces, which is especially common in a very stiff system. Hence, we further empirically adapt the elastic stiffness to compensate for the energy deviation and recover it after node resampling, which helps to suppress the jittering artifacts in practice.

The major technical contributions can be summarized as:

- Unify both Lagrangian and EoL nodes in representation to extend the discrete configuration space, and formulate a constrained optimization to resolve numerical degeneracy based on a moving mesh discretization of the Cosserat rod.
- Propose to constrain material points to be equally spaced with respect to rod curvature by an additional energy to govern the sliding of material points. In each time step, such a strategy can flexibly adapt the sample points to reproduce local small scale geometry.
- Incorporate *h*-adaptive idea using some practical strategies to dynamically resample nodes. Based on that, we further empirically adjust the energy stiffness whenever there is node collapsing to suppress jittering artifacts probably arising in a stiff system.

2. Related Work

Elastic rod simulation The modeling and simulation of one-dimensional flexible structure (i.e. an elastic rod) are important problems in mechanics. Because of the existence of significant twist and bend, rod simulations are often more complicated than solids or thin-shells. To develop a basic model of a rod, its centerline, represented by a material curve, must be capable of resisting bending and torsion. From a historical perspective [O'R08], Kirchhoff first presented a rod theory capable of modeling bending and torsion [Kir59], and is used to simulate the inextensible 1D elastic object in computer graphics, such as hair [BAC*06] and strands [BWR*08].

In the early 20th century, the Cosserat brothers presented a formulation of Kirchhoff's rod theory [CC07, CC09] using what we now call *directors*, therefore the rod theory we discuss is often considered as an example of the Cosserat rod theory. Since Pai [Pai02] introduced the Cosserat model simulation to the graphics community, the model has been studied and developed for years. Spillmann et al. [ST07] developed a Cosserat rod model that uses quaternion to represent material frames, penalize the frame to conforms to the rod's orientation, and measure the deformation on off-tangent directions. An efficient material frame representation based on parallel transport was also presented in [BWR*08] and [BAV*10].

To compromise simulation accuracy and computational cost, some fast simulation methods have been developed by the graphics community for interactive applications. Umetani et al. [USS14] developed a Cosserat rod simulation in the position based dynamics (PBD) framework by introducing ghost points. Later, Kugelstadt et al. [KS16] presented a PBD-based approach that discretizes material frame orientation using quaternions without ghost points. More recently, Soler et al. [SMSH18] proposed a projective dynamics method that further accelerates the simulation of Cosserat rods.

***h*-, *r*- adaptive simulation** Many adaptive techniques have been employed in physics-based simulation including solid simulation

and Eulerian and Lagrangian fluid simulation. Here, we briefly introduce previous studies on adaptive simulation on structural elements. Please refer to the paper [MWN*17] for a comprehensive survey. Grinspun et al. [GKS02] presented an adaptive framework by refining the basis functions rather than changing the discretization mesh. In contrast, the *h*-adaptive method can physically adjust the mesh size and dynamically remesh regions of interest for improved simulation accuracy. In graphics, Narain et al. [NSO12] presented a framework to adaptively simulate clothing, which is further extended to solve folded papers [NPO13] and view-dependent adaptive cloth simulation. As for elastic rods, Spillmann et al. [ST08] proposed an adaptive contact model to simulate knots. To summarize, *h*-adaption is an intuitive approach to improve simulation accuracy by having more degrees of freedom.

Unlike *h*-adaption, *r*-adaptive method (a.k.a. moving mesh method) tries to find a "smart" way to reallocate degrees of freedom without having additional ones, hence it can get rid of possible fictitious vibration caused by *h*-adaption. This kind of methods has been studied for years [BHR09, HR10], which turns out an effective approach to accurately solve violent time-variant systems. As a variant of the *r*-adaptive method, Eulerian-on-Lagrangian (EoL) method was introduced by pioneering works [SJLP11, SLNB10], which combined *h*-adaption techniques to simulate elastic cables in contacts smoothly by allowing material points to continuously slide inside the parametric domain. Later, this idea was further extended to simulate a variety of objects, such as skin [LSNP13], elastic and elastoplastic deformable solids [FLLP13], continuum-level cloth [WPLS18] and yarn-level cloth [CLMMO14, CLMO16]. Given the numerical singularity introduced by Eulerian coordinates, these methods either try to maximize the Lagrangian velocity, which should amount to restricting the Eulerian velocity in some senses to make material points as static as possible, or apply elastic forces to drive their dynamics as rigid as possible. However, these approaches are arguably limited in adaptivity brought by Eulerian coordinates, seemingly going in an opposite way as *r*-adaption intends. Targeting at numerical conditioning, a recent work [SBRBO20] introduced a special type of nodes to avoid infinitely large stiffness resulted from tiny segments, which can be also empirically solved by collapsing tiny segments in each step.

3. Cosserat Model under Moving Mesh Discretization

For an elastic rod Ω with rest length L, arc-length parameterization gives the material (Eulerian) coordinate $u(p) \in [0,L]$ for any point $p \in \Omega$. Its shape can be characterized as a function $\mathbf{x} : [0,L] \to \mathbb{R}^3$. When applying common piecewise linear discretization, rod shape is discretized as sequential segments $\mathbf{e}_i = [\mathbf{x}_i, \mathbf{x}_{i+1}]$, and each node $\mathbf{x}_i$ has its material coordinate u_i on the rod. Any deformation on the material point inside the segment $[u_i, u_{i+1}]$ can be linearly interpolated via

$$\mathbf{x}(u) = \frac{u - u_{i+1}}{u_i - u_{i+1}}\mathbf{x}_i + \frac{u_i - u}{u_i - u_{i+1}}\mathbf{x}_{i+1}, \quad u \in [u_i, u_{i+1}]. \tag{1}$$

To simulate the rod dynamics, we need to specify the kinetic energy and potentials of the system to govern its dynamical evolution, which involves the integration over the velocity field $\dot{\mathbf{x}}(u)$ and deformation gradient field

$$\mathbf{F} = \nabla_u \mathbf{x} \in \mathbb{R}^3. \tag{2}$$

Taking the above piecewise linear discretization, any integration can be evaluated as the summation over all segments.

In traditional FEM, the discretization itself is regarded as static, which means the material coordinate u of each node is a constant. Though making the equation of motion much simpler in both mathematical formulation and computation, it sometimes could bring in severe artifacts, such as the well-known over-stiffening problem or even catastrophic locking phenomena arising in over-constrained system. In contrast, moving mesh discretization provides each node with a changeable material coordinate, generally described as a time-variant function. However, this extension immediately imposes a question: once the material point can be moved, how should they get moved?

Besides, the velocity of any material point is not only about the time derivative of its spatial coordinate $\mathbf{x}$ now, but is also about associated Eulerian coordinate u, which can be evaluated as the total time derivative of the spatial coordinate [SJLP11]:

$$\mathbf{v}(\mathbf{x}, u, \dot{\mathbf{x}}, \dot{u}) = \dot{\mathbf{x}} - \nabla_u \mathbf{x} \cdot \dot{u}. \tag{3}$$

As discussed in previous works [FLLP13], this will make the dynamical system degenerate and possibly cause an unstable simulation if not handled properly. Hence, another quick question arisen is that how to address such numerical degeneracy effectively.

As we will see, these two questions eventually relate to how to properly constrain the motion of those material points. In the following content of this section, we will show how to address the above two challenges on the Cosserat model, and later demonstrate how to integrate *h*-adaptive FEM idea of dynamical resampling for an efficient rod simulation in the next section.

3.1. Cosserat model in brief

Given the material point velocity by Equation (3), the translational kinetic energy is formulated as

$$T_k(\mathbf{x}, u, \dot{\mathbf{x}}, \dot{u}) = \frac{1}{2}\int_0^L \rho\pi r^2 \|\mathbf{v}\|^2 du, \tag{4}$$

where r is the radius of cross section of the rod and ρ is the material density. We should notice that now computing the integral over the range $[0,L]$ relies on a time-variant discretization defined by the Eulerian coordinates $\{u_i\}$. Since the velocity is configuration dependent, the mass matrix will be consequently relying on both Lagrangian and Eulerian coordinates, and becomes time-variant in our simulation. Similarly, the potential also relates to both Eulerian and Lagrangian coordinates. For instance, to keep a rod as inextensible as possible, we will have a potential to penalize rod stretching written as

$$V_s(\mathbf{x}, u) = \frac{1}{2}\int_0^L k_s(\|\mathbf{F}\| - 1)^2 du, \tag{5}$$

where k_s is the stretching stiffness.

However, if only node positions are taken into account, we cannot model the deformation along the normal and binormal directions of a curve. According to the Cosserat theory, this can be done

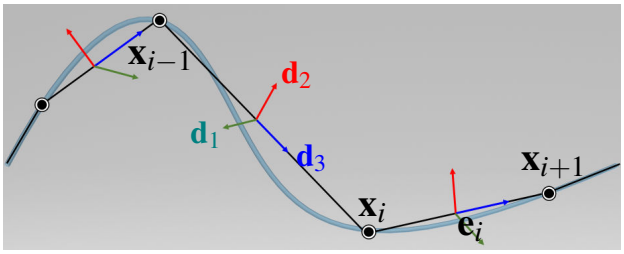


Figure 2: ***Discretization of Cosserat rod****. The discrete configuration of the Cosserat rod is determined by both spatial position* $\{\mathbf{x}_i\}_i$ *and material frames defined on each segment* $\{\mathbf{e}_i\}_i$*, which is composed of tangential, normal and binormal vectors* $\mathbf{d}_3, \mathbf{d}_2, \mathbf{d}_1$*. The material coordinate* $\{u_i\}_i$ *for each sample point is no longer static in our moving reference discretization.*

by tracking a dynamic orthonormal material frame $\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3\}$ on each line segment $\mathbf{e}_i$ (see Figure 2). In practice, the orthonormal frames are encoded as quaternions $\{\mathbf{q}_i\}_i$ for computation, simply following the treatment in [ST07].

With such a time-variant material frame field $\mathbf{q}(t)$, the rotational kinetic energy is calculated as

$$T_r(\mathbf{q}, \dot{\mathbf{q}}, u) = \frac{1}{2}\int_0^L \|\mathbf{J}\boldsymbol{\omega}(\mathbf{q}, \dot{\mathbf{q}})\|^2 du, \tag{6}$$

where $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the inertia tensor and $\boldsymbol{\omega} \in \mathbb{R}^3$ represents the angular velocity which is computed by projecting the frame velocity $\dot{\mathbf{q}}$ into the local frame expanded by $\mathbf{q}$. Similarly, we can define elastic potential of the material frame which governs bending and twisting of a rod as

$$V_b(\mathbf{q}, u) = \frac{1}{2}\int_0^L \|\mathbf{K}(\boldsymbol{\varepsilon}(\mathbf{q}) - \bar{\boldsymbol{\varepsilon}})\|^2 du, \tag{7}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^3$ is the strain of the frame (a.k.a. Darboux vector) and $\bar{\boldsymbol{\varepsilon}}$ conforms to the natural bending and torsion of the rod based on its original geometry, mathematically evaluated as the projection of the spatial derivative $\nabla_u \mathbf{q}$ to the local frame $\mathbf{q}$. For simplicity, we just simulate rods that are straight in their rest states, where $\bar{\boldsymbol{\varepsilon}}$ is simply zero. Besides, the stiffness $\mathbf{K} \in \mathbb{R}^{3\times3}$ encodes the resistance to frame deformation in different directions.

In addition, we have constraints to couple the spatial coordinate and the material frame to enforce that the tangent deformation of elements should coincide with the material frame in the same direction, i.e.,

$$C_p(\mathbf{x}, u, \mathbf{q}) = \frac{\mathbf{F}}{\|\mathbf{F}\|} - \mathbf{d}_3(\mathbf{q}) = \mathbf{0}, \tag{8}$$

where $\mathbf{d}_3$ represents the tangent vector of the frame $\mathbf{q}$ as illustrated in Figure 2. Once the length is preserved, this constraint can be simply approximated as:

$$C_p(\mathbf{x}, u, \mathbf{q}) = \mathbf{F} - \mathbf{d}_3(\mathbf{q}) = \mathbf{0}. \tag{9}$$

At last, since $\mathbf{q}$ itself represents a quaternion, it must have unit norm, so that we have

$$C_q = \|\mathbf{q}\| - 1 = 0. \tag{10}$$

The technical details on computing above energies in discrete settings will be given in appendix A and readers can also refer to [ST07] for a thorough discussion. With our moving reference discretization, we should always keep in mind that after discretization, all relevant energy calculation is calculated by summing up the contributions from each dynamically updated parametric range on the reference domain, which are essentially functions of both Eulerian and Lagrangian coordinates.

3.2. Constrain the motion of Eulerian coordinates

The additional Eulerian coordinate does offer a larger space to adapt the discretization, but at the same time it can create problems of numerical degeneracy. Before diving into the details on addressing such an issue, we firstly show how those Eulerian coordinates can become redundant in representing dynamic states, and thus make the numerical system rank-deficient and unstable to solve if not handled properly.

When taking material coordinate into account as a part of generalized coordinates, same kinetic status can be achieved via either Lagrangian or Eulerian motion up to a certain transformation [FLLP13]. Numerically, it is revealed by the fact that there should exist non-zero perturbation around a given state that keeps energies invariant. As a consequence, the mass and stiffness matrices can be singular, leading to multiple feasible solutions when integrating the equation of motion.

To clearly expose the problem, we can check the mass which is evaluated as the second derivative of the kinetic energy w.r.t. the generalized velocity to demonstrate the redundancy caused by those additional Eulerian coordinates. In discrete settings, the generalized mass matrix is assembled by small blocks $\mathbf{M}_i$ computed over each line segment $\mathbf{e}_i$:

$$\mathbf{M}_i(\mathbf{x}_i, \mathbf{x}_{i+1}, u_i, u_{i+1}) = \frac{\rho\pi r^2(u_{i+1} - u_i)}{6}\begin{bmatrix} 2\mathbf{I}_{3\times3} & \mathbf{I}_{3\times3} & -2\mathbf{F} & -\mathbf{F} \\ \mathbf{I}_{3\times3} & 2\mathbf{I}_{3\times3} & -\mathbf{F} & -2\mathbf{F} \\ -2\mathbf{F}^T & -\mathbf{F}^T & 2\mathbf{F}^T\mathbf{F} & \mathbf{F}^T\mathbf{F} \\ -\mathbf{F}^T & -2\mathbf{F}^T & \mathbf{F}^T\mathbf{F} & 2\mathbf{F}^T\mathbf{F} \end{bmatrix}, \tag{11}$$

where the piecewise constant deformation gradient $\mathbf{F}$ is evaluated as $(\mathbf{x}_i - \mathbf{x}_{i+1})/(u_i - u_{i+1})$. It is easy to find that above mass matrix is always singular, whose nullspace is spanned by two kernel vectors $[\mathbf{F}^T, \mathbf{0}_3^T, 1, 0]^T$ and $[\mathbf{0}_3^T, \mathbf{F}^T, 0, 1]^T$, meaning that as long as the element is undergoing an infinitesimal deformation located in this kernel space, the kinetic energy will never change although both Lagrangian and Eulerian nodes can be notably moved. Similarly, such degeneracy also happens to stretching potential. For instance, the material points could always move along with Lagrangian ones to cancel out the actual deformation once the segment length is precisely preserved in both deformed and reference configuration. Consequently, the rod shape can be significantly distorted without doing any work, eventually producing ill-defined problems and thus unstable simulations.

Knowing the source of degeneracy, we should note that the method to address this problem is far from unique. One can quickly add some certain constraints on either Lagrangian or Eulerian coordinates to remove the nullspace, and only has to ensure the

equation of motion is held to obtain valid dynamics. Existing methods either try to strictly fix the Eulerian coordinates such as traditional FEM method, or enforce them as static as possible [FLLP13, CLMMO14]. Though being able to move Eulerian coordinates to some extent by collision response or some fictitious friction forces, these methods arguably pose too strong restrictions on the motion of material points, which somehow goes an opposite way in principle as moving mesh method intends to do.

In our method, the material point can be flexibly moved to offer better *r*-adaption to rod geometry. Among all possible technical solutions, we make a choice standing on the point of approximation error by FEM and constrain the motion of material points based on a simple prior: *region having a larger curvature should be more densely sampled*. To realize this target, we first construct a density field $\theta : [0, L] \to R^+$ that reflects curvature distribution. Used as a monitor function to control the re-distribution of material points, our method forces the samplings to be equally spaced with respect to this density function. More precisely, given this field $\theta(u)$, the additional regularization term to govern the Eulerian motion can be formulated as an optimal transport like functional, i.e.,

$$R(\mathbf{u}) = \sum_i \int_{\Omega_i} \theta(u) \|u - u_i\|^2 du, \tag{12}$$

where Ω_i is the Voronoi region of node u_i, and in 1D it simply amounts to $[(u_{i-1} + u_i)/2, (u_i + u_{i+1})/2]$. This function is also known as the variational form of the centroid Voronoi tessellation (CVT) [LWL*09] for meshing, where u_i is also referred to a "site" of the associated Voronoi cell. When the curve is discretized by a chain of piece-wise linear segments, the curvature will be only concentrated on each individual vertex and is vanishing elsewhere. Thus, for each vertex the density function is evaluated per node as a quadratic function with the value range of $[\theta_{min}, \theta_{max}]$, namely

$$\theta_i = \theta_{min} + \theta_{max} \left(\frac{\beta_i}{\pi} \right)^2, \tag{13}$$

where each β_i is the turning angle on that vertex reflecting the change on the tangent direction, which can be seen as locally integrated curvature, and is normalized to $[0, 1]$ then. Compared to a simple linear mapping, using quadratic one can accelerate the material points to slide to get aggregated from flat to curved regions. Note that this mapping is not unique, one can flexibly change this function to customize the "acceleration" of material point sliding. In our experiments, the contrast between θ_{min} and θ_{max} is generally set as 10^3.

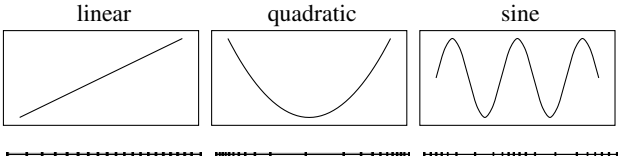


Figure 3: ***Density driven resampling****. Given the density function (top), the original equally spaced nodes will be redistributed by optimizing energy by Equation* (12) *with respect to the density change (bottom).*

In 2D and 3D, the smoothness of CVT energy is C^∞ if the combinatorial structure of the Voronoi diagram does not change, otherwise it relates to the smoothness of the density function [LWL*09]. In our 1D case, the rod shape is generally smooth so does the quadratic mapping, and due to the existence of inextensible energy it is not easy for reference elements to get inverted. Consequently, our proposed regularization function does not introduce visible numerical difficulties such as falling into undesired local minima or being unable to converge as observed in our experiments. By treating the density field to be piece-wise constant over dual cells at each timestep (Equation (13)), Equation (12) can be accordingly approximated as

$$\begin{aligned} R(\mathbf{u}) &= \sum_i \int_{\frac{u_{i-1}+u_i}{2}}^{\frac{u_i+u_{i+1}}{2}} \theta_i \|u - u_i\|^2 du \\ &= \sum_i \frac{\theta_i}{24} \left[(u_{i+1} - u_i)^3 + (u_i - u_{i-1})^3 \right], \end{aligned} \tag{14}$$

which is mathematically a cubic function in all Eulerian coordinates. For a boundary node, the point next to it is mirrored to form a virtual boundary to compute the integral. Some different density functions and the resulting sampling distribution after optimizing Equation (12) are depicted in Figure 3.

3.3. Rod dynamics

In this subsection, we will show that the time stepping of Cosserat rods with moving mesh discretization can be formulated as a constrained optimization problem. Firstly the equation of motion (EoM) with the generalized coordinates will be derived. For simplicity of exposition, we only discuss the nodal position part, i.e., $\mathbf{x}$ and u. As for material frame $\mathbf{q}$, its associated equation of motion is provided in Appendix B under a similar treatment as positional variables.

With the extended Eulerian coordinate, the dynamic state of a rod can be described by the generalized coordinate $\mathbf{r} = [\mathbf{x}^T, \mathbf{u}^T]^T$ and its velocity $\dot{\mathbf{r}}$. By constructing the system Lagrangian $\mathcal{L} = T_k - V_s$ and applying the well-known Euler-Lagrange equation, the equation of motion can be finally derived as

$$\mathrm{EoM}(\mathbf{r}, \dot{\mathbf{r}}): \quad \dot{\mathbf{M}}(\mathbf{r})\dot{\mathbf{r}} + \mathbf{M}(\mathbf{r})\ddot{\mathbf{r}} - \frac{\partial T_k}{\partial \mathbf{r}} + \frac{\partial V_s}{\partial \mathbf{r}} = \mathbf{g}, \tag{15}$$

where $\mathbf{M}(\mathbf{r})$ is the position-dependent mass and $\mathbf{g}$ is the external force. As discussed in Section 3.2, the Eulerian coordinates need to be further constrained in addition to the equation of motion for an adaptive redistribution. Therefore, we can formulate the time stepping from nth step to $n+1$ as a constrained optimization problem:

$$\begin{aligned} \underset{\mathbf{r}^{n+1}}{\arg\min} \quad & R(\mathbf{u}^{n+1}; \boldsymbol{\theta}^n) \\ \text{s.t.} \quad & \mathrm{EoM}(\mathbf{r}^{n+1}; \mathbf{r}^n, \dot{\mathbf{r}}^n) = 0, \\ & \mathrm{EoM}(\mathbf{q}^{n+1}; \mathbf{q}^n, \dot{\mathbf{q}}^n) = 0, \\ & C(\mathbf{r}^{n+1}, \mathbf{q}^{n+1}) = 0, \\ & W(\mathbf{r}^{n+1}, \mathbf{q}^{n+1}) \geq 0, \end{aligned} \tag{16}$$

where the regularization term refers to Equation (14), and $\boldsymbol{\theta}^n$ denotes the density vector from the nth step and is evaluated at each

point by Equation (13). In addition, the catenation of all equality constraints in dynamical system is denoted by C, such as boundary fixing conditions, coupling constraints C_p (Equation (9)), quaternion constraints C_q (Equation (10)), etc. The other symbol W represents for all inequality constraints, mainly consisting of all collision response constraints (see Section 5.2).

To have a stable simulation, we use the implicit Euler method to integrate the equation of motion. However, strictly applying the implicit Euler scheme to Equation (15) can be very complicated since mass is constantly changing. For simplicity, we technically take a combination of implicit and explicit treatment on the unknowns appearing in different places in the EoM. For instance, the resulting time-stepping equation for $\mathbf{r}^{n+1}$ can be approximated as

$$\mathbf{M}(\mathbf{r}^n)\frac{\mathbf{r}^{n+1}-\mathbf{r}^n-h\dot{\mathbf{r}}^n}{h^2}+\frac{\partial V_s}{\partial \mathbf{r}}(\mathbf{r}^{n+1})=\mathbf{g}, \tag{17}$$

where h is the time step generally set as 8×10^{-3}s in our experiments. Note that two terms, i.e., $\dot{\mathbf{M}}(\mathbf{r})\dot{\mathbf{r}}$ and $-\partial T_k/\partial \mathbf{r}$ in Equation (15) are dropped in above equation, similar to the approach done in [SJLP11]. The intuition here is that as long as the motions of material points are not violent, these two terms are basically subtle and thus can be safely neglected. Such a treatment makes the evaluation much simpler while does not harm physical realism too much. We will give relevant algorithmic details on solving problem (16) in Section 5.2.

4. Adaptive Resampling

Though being adaptive to rod geometry by redistributing material points under the government of the CVT regularization, the discretization can be still limited to capture sharp changes on dynamic states, which are often caused by some unpredictable instantaneous impulses such as a sudden contact. To get over the limitation of using a fixed number of elements, we further equip our moving reference discretization with h-adaptivity, which allows to insert or merge sampling nodes online to balance physical accuracy and numerical efficiency, a crucial feature especially indispensable for contact intensive environments.

In this section, we will briefly introduce the geometric criteria for node insertion and deletion. Since similar topics have been greatly explored in graphics community and we just explain necessary technical details to make the paper self-contained, without introducing too many novel observations or techniques. It is worth mentioning that once the nodes have been removed, our method will accordingly adjust the elastic stiffness to suppress possible jittering probably showing up in a stiff system, and the modified stiffness will be gradually restored in following simulations.

4.1. Geometric criteria

To begin with, we will explain the criteria to determine when and where to insert or delete sample points.

Node insertion One typical scenario that needs to insert additional nodes is when a rod is contacting a sharp edge on rigid bodies (see Figure 4 (a)). The initial discretization likely cannot have a sample point at the collision point, which will inevitably produce spurious jittering in the simulation. To address this problem, we choose to insert a node exactly at the collision point whenever the rod is hitting a sharp edge of obstacles. Also, rods can contact themselves, for instance in a process of forming knots. Similarly, we will insert a node for each colliding segment at the self-collision point whenever two segments are hitting each other (see Figure 4 (b)). Lastly, the rod is of one-dimensional structure in geometry, and this fact makes it very easy to bend and twist, often deforming into shapes with sharply varying curvature. In this situation, the material points can automatically slide towards curved regions from flat regions where segment length can accordingly become too long. Thus, to avoid a too long element, a node will be inserted at the center of a line segment whenever its length is exceeding a given tolerance relative to the initial average element length (see Figure 4 (c)). On the one side, this prevents having too many long elements on flat regions, while it is also a source to inject more nodes, which is in charge of providing sufficient DoFs to improve the asymptotic accuracy of r-adaption.

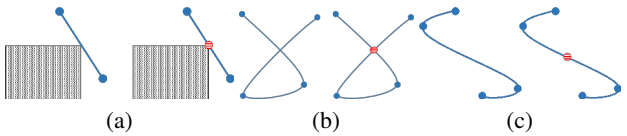


Figure 4: ***Node insertion****. When a line segment is contacting an edge of a rigid body, or it hits another rod segment, or its length over exceeds a tolerance, additional nodes will be inserted.*

Node reduction Suppose that a rod is hitting an object and if the collision lasts for a period of time, constantly inserting new nodes around collision points could produce a large number of samplings residing in a small contact region. A similar situation could also happen to highly curved region due to node sliding. As system advances, these tiny segments are probably no longer critical to capture sharp features on a rod, but instead might make the system infinitely stiff [SBRBO20]. Therefore, considering both numerical conditioning and computational efficiency, we technically collapse all tiny line segments in practice as [SJLP11] did, as long as the segment length is smaller than a given tolerance relevant to the length scale used for collision detection.

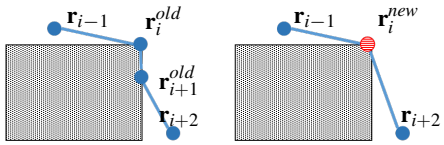


Figure 5: ***Node collapsing****. Our node collapsing strategy is aware of collision constraint and its historical information, which empirically alleviates jittering artifacts produced by naive averaging for node reduction. In this figure, r_i^{old} is an old collision point that has been staying at the collision point for a while, hence it has the highest priority. Node r_{i+1}^{old} is an ordinary one so it will be merged into r_i^{old} to generate a new node r_i^{new}.*

When trying to collapse two neighboring nodes, the position of

the new collapsed node has to be decided anyway. As one can imagine, a naive averaging will inevitably produce many undesired jittering artifacts. To remedy this issue, [WPLS18] merges the nodes based on their attached labels, either as EoL or Lagrangian. Here we inherit a similar idea but perform node collapsing in a slightly different way. Our strategy is based on the historical constraint information recorded by each node. If one of two nodes had edge collisions in the last time step, it will have the highest priority for merging, which means that its neighboring nodes will be always merged into it. This kind of node is actually critical for exactly maintaining the corner shape of a rod passing a sharp edge, which also makes the remeshing process more stable. In contrast, if a freshly inserted node was assigned a higher priority, the mesh would jitter drastically in contact areas. Additionally, if a node only has edge collisions in the current step, it will be assigned a medium priority. Lastly, the nodes without any contacts in two consecutive steps have the lowest merging priority. Although our priority scheme cannot fully eliminate jittering for arbitrary collapsing, it generally works well in practice. Figure 5 gives an illustration.

If two nodes with the same priority are going to be merged, e.g. represented by $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ respectively, the final position of the collapsed node is obtained by

$$\mathbf{r}_i^{new} = \frac{\beta_i}{\beta_i + \beta_{i+1}} \mathbf{r}_i^{old} + \frac{\beta_{i+1}}{\beta_i + \beta_{i+1}} \mathbf{r}_{i+1}^{old},$$

where β stands for the turning angle, and such a merging scheme is adapted to the local geometry as a way to identify nodal importance for node collapsing. Notice that the denominator in above equation can be zero, which means both nodes are in a straight line, in which case the new node will be simply laid at the middle point.

4.2. Physical model adaptation

Node insertion As above, the node insertion mechanism can be triggered for three reasons: collisions with sharp features, self collisions, and long segments. After getting the position of an inserted node, both Lagrangian and Eulerian velocity of this node is computed by simple linear interpolation. To avoid increasing bending and twisting energy, the initial frame strain $\bar{\boldsymbol{\varepsilon}}$ at the new node is set to be zero, and the original material will be copied to the splitted segments. In this way, there will be no potential energy increase in bending and twisting when there are node insertions.

Node reduction Compared to node insertion, energy adaptation for node reduction is more involved. As we have pointed out, every change to the discretization will take a risk of jittering. Merging two nodes into a new one will obviously change the local configuration and result in a loss, which could cause significant energy deviation in potentials, especially severe for the stretching part due to the triangular inequality. As a consequence, the ghost stretching forces introduced by node collapsing can be very large, which likely leads to drastic jittering or even makes the system matrix indefinite. One of the approaches to reduce the jittering is proposed by [NPO13] which modifies the node positions to avoid a large jumping in forces. Instead, we propose to adaptively adjust the stiffness to alleviate this problem without changing node positions for simplicity.

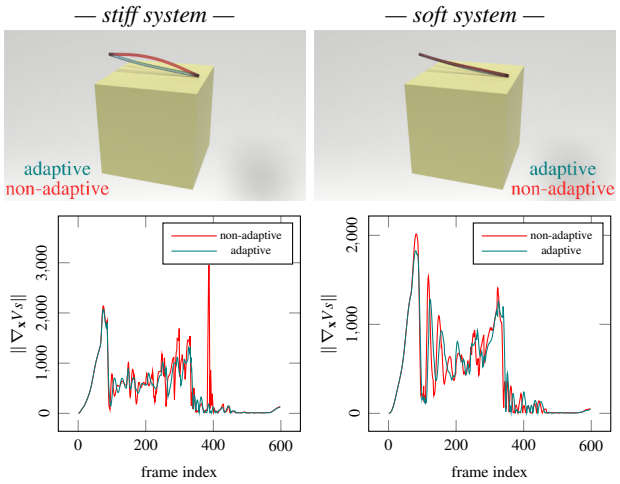


Figure 6: ***Stiffness adjustment****. Dragging a rod along a cube edge will trigger the mechanism on node insertion and reduction. For a stiff system (left), merging nodes can introduce a drastically large ghost force as showing up around the 400th frame and cause spurious jittering visually. With our strategy to adjust the stiffness after node reduction, such an artifact is alleviated in terms of both stretching force magnitude and visual smoothness. Moreover, such adaption still behaves consistently for a soft system (right), without a notable deviation from the non-adaptive version as a reference.*

Taking Figure 5 for an illustration, after collapsing the tiny elements, the original three line segments will be merged into two. Our idea to adapt the stiffness is to keep the stretching energy of the remained elements unchanged after collapsing. For instance, denote the old and new energy values calculated with original stiffness k_s on segment $[\mathbf{r}_{i-1}, \mathbf{r}_i]$ as $V_s(\mathbf{r}_{i-1}, \mathbf{r}_i)$ and $V_s'(\mathbf{r}_{i-1}, \mathbf{r}_i)$ respectively. To keep the stretching energy unchanged, we can adjust the stiffness of this segment to the ratio between the old and new energy values, i.e. $V_s(\mathbf{r}_{i-1}, \mathbf{r}_i)/V_s'(\mathbf{r}_{i-1}, \mathbf{r}_i)$. If there is no stretch on this segment, it is not necessary to adjust the stiffness. Also, to prevent the adapted stiffness from being too soft or too stiff, we will further clamp the stiffness in a given range in practice. The range is generally bounded by 0.1 and 10 times of the original stiffness in our experiments. Other strong energies in the system can be also treated in this way if node collapsing is occurring. The effect of adapting the stretching stiffness is present in Figure 6.

To keep the mechanical consistency after node resampling, the modified stiffness will be restored to its original value in the upcoming frames. In each frame, the algorithm will iterate over each segment without collisions and recover its stiffness if necessary to an intermediate value at a prescribed recovery speed.

As we can see, while *h*-adaption could cause some undesired artifacts indeed and there seems no perfect way to deal with such issues, it still greatly improve the asymptotic accuracy of the discretization on a positive side. Instead, pure *r*-adaptive method is popping-free but its accuracy is generally limited by its initial discretization granularity. Therefore, by combing both *r*- and *h*-adaption, the expressivity of our method has been significantly

strengthened so that it can capture complex dynamics with only limited DoFs.

5. Results

In this section, we first provide a sketch of our simulation pipeline for each single time step, and then give the implementation details on handling collisions and solving the constrained optimization. Next, several testing cases are present to demonstrate the effectiveness of our proposed method.

5.1. Simulation pipeline

The simulation pipeline for each timestep is summarized in Algorithm 1. At the beginning of each step, we update the bounding volume hierarchy tree and detect all collisions. After that, according to the collision registration, we resample the material nodes for h-adaption. Next, we compute internal and external forces, stiffness and mass matrices, identify constraints, construct the equation of motion and finally solve the problem 16 to update dynamic states of the system. It should be noted that in this process, we do not label or tag nodes being EoL or Lagrangian as [WPLS18] did, but we do distinguish nodes based on their types of associated constraints for node merging as mentioned.

Algorithm 1 Time stepping pipeline

1: **while** simulating **do**
2: Collision detection. (§ 5.2)
3: Resample. (§ 4)
4: Compute force and apply constraint. (§ 3.1)
5: Solve problem (16) to update velocities and positions. (§ 3.3)
6: **end while**

5.2. Implementation details

Collision handling The collision handling includes two steps: collision detection and collision response. As for detection, we simply adopt the methods proposed by [BFA02]. An axis-aligned bounding box hierarchy is used to accelerate the detection of proximity of possible collision primitives, which is built up-bottom once at the beginning of each step using the geometry of the objects. Checking proximity of geometric primitives can be atomized into two fundamental tests, i.e. vertex-face and edge-edge intersections. If the distance between elements is below a tiny rounding tolerance, we will register a collision for these corresponding elements.

As for collision response, we mainly follow the methods used in [WPLS18]. Existing contacts can be resolved by adding inequality constraints to the optimization problem for time stepping. Technically, the constraints can be classified into three types:

- Vertex-face constraint: if the vertex $\mathbf{x}_i$ collides with a triangular face of a rigid body with a normal vector $\mathbf{n}$, we will constrain its Lagrangian velocity such that $(\mathbf{v}_i - \mathbf{v}_r) \cdot \mathbf{n} \geq 0$, where $\mathbf{v}_r$ is the velocity of the rigid obstacle.
- Vertex-edge constraint: because we dynamically insert nodes at collision points, the collision between a rod segment and an obstacle edge can be turned into vertex-edge constraints for rigid body contacts. Specifically, for a node $\mathbf{x}_i$ contacting a solid edge, the associated vertex-edge constraint can be transformed into two vertex-face constraints assembled on the two neighboring faces of that edge.
- Edge-edge constraint: above two types of constraints are used for handling collisions with sharp features on rigid bodies. For self contact, vertex-edge constraints are not enough to avoid self-penetration in our experiments, because there could be large amounts of collisions from different directions happening to one single segment. Thus, we use edge-edge constraints for self collisions. In our implementation, we apply inequality constraints on two ending nodes of contacting edges in order to separate two contacting edges. Related details are referred to [BFA02].

Solving constrained optimization In practice, we technically make some approximation when solving the constrained optimization given by Equation (16) for efficiency consideration. Some of the equality constraints are relaxed as soft penalties in our current implementation with a large weight, such as the coupling constraint given in Equation (9), which follows the same treatment applied by [ST07]. Besides, the EoM as a constraint can be also relaxed by sacrificing a bit of physical accuracy to interchange computational efficiency. After making EoM constraint as penalty functions, the objective is optimized subject to other inequality constraints (mainly resulted from collisions) by quadratic sequential programming. We simply take the implementation from Mosek [ApS19] code-wise, and relevant algorithmic details can be referred to [ART03]. Since the weight for EoM penalty is generally set 10^6 times larger than that for regularization term, we still observe enough dynamics and do not find any notable artifacts by conducting above relaxation. As for the unit length constraint imposed on quaternion (Equation (10)), a simple post normalization at the end of each step is already enough to generate plausible results. Considering potentially intensive collisions in simulation scenarios, our prescribed time step is not set that large in order to prevent undesired penetrations. As a consequence, one or two times of iterations for each time step is mostly enough.

5.3. Simulation testings

We implemented our simulation algorithm in C++ and ran the testings on a desktop with an Intel Core i7-6700HQ CPU@2.60 GHz and 8 GB of RAM. We use Eigen library [GJ*10] for linear algebra and Mosek [ApS19] for quadratic programming to solve the time stepping. All the testing cases present in this paper can run in real-time performance. In this section, we will present some simulated dynamics to demonstrate the effectiveness of our method. The animation can be seen in the accompanying video.

Twisting Due to the coupling of bending and twisting, if both ends of the rod are pulled in opposite directions and twisted simultaneously, the rod will form knots near the middle part. As shown in Figure 7, from the close-up figures and the bottom plots of the reference mesh we can see that the nodes are re-distributed more compactly in regions with high curvature values, which makes the simulated dynamics more smooth and plausible. Also, the second column clearly shows that additional samplings can be automatically inserted if there are self-collisions. And in the last column, it turns out that our method can well reproduce complex geometric details of multiple knot loops when the rod is getting excessively twisted.

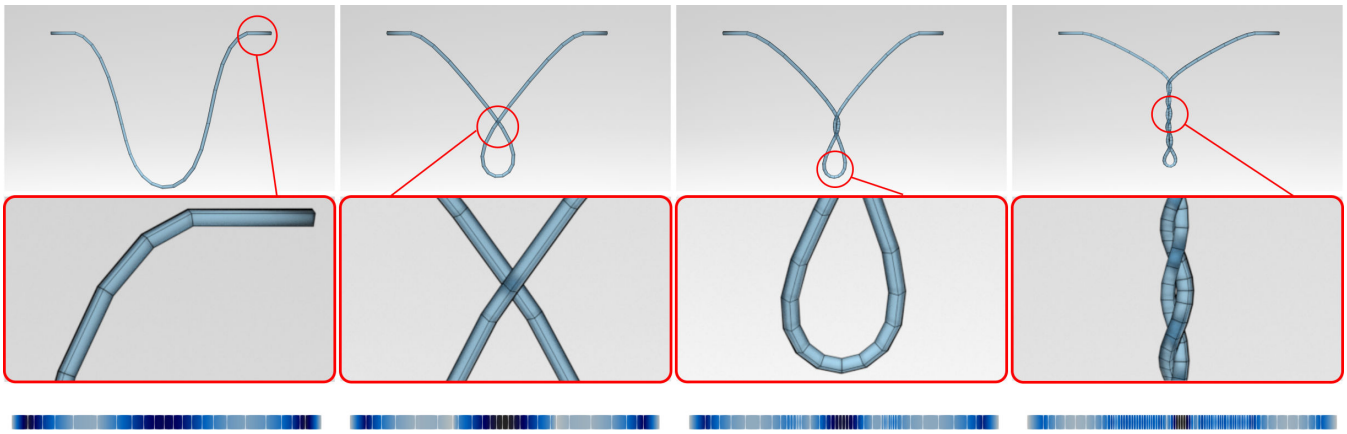


Figure 7: ***Twisting with self-contact***. *Cosserat rod can produce knots when both ends are pulled in opposite directions and twisted simultaneously. As the system evolves, the rod shapes become increasingly complex due to knotting, forming a non-uniform curvature distribution. With our r-adaptive strategy, the original uniformly spaced reference mesh can slide or split to adapt to the local curvature as well as contacts which is illustrated in the close-up figures. The bottom row plots time-variant reference mesh changing with rod geometry, where deeper color corresponds to a larger curvature value.*

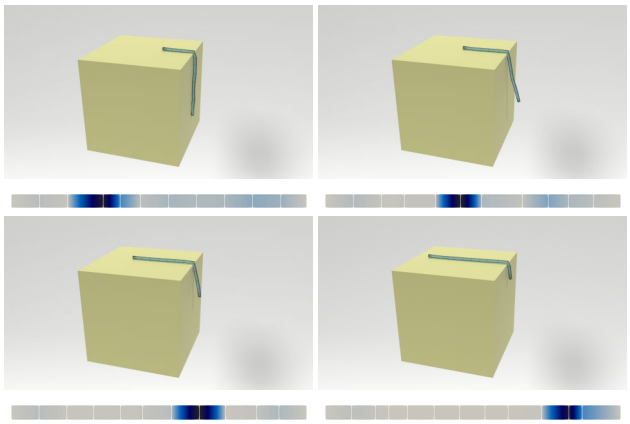


Figure 8: ***Rigid body contact***. *When dragging one end of a soft rope to make it slide along a sharp edge of an obstacle, our method will always have a node on the collision point, which ensures that the velocities can be smoothly transited along the collision edge with the help of the Eulerian coordinates. Also, since the collision point has the highest curvature, samplings will be moved towards the collision area, which further refines the dynamical smoothness.*

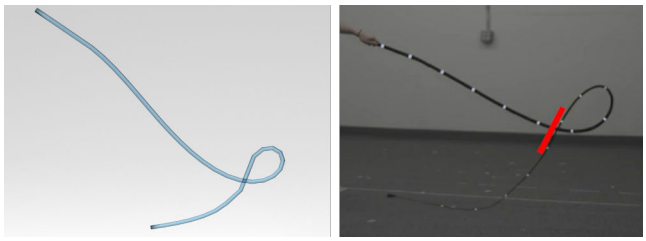


Figure 9: ***Crack of a whip***. *By moving the material points according to the curvature distribution, our method can simulate a cracking whip with a loop of varying size propagating along the rod.*

Rigid body contact By incorporating *h*-adaption, our method can accurately handle collisions, where there are always sharp changes on dynamical states which are generally challenging for the initial discretization to capture. As illustrated by Figure 8, when the rod is contacting with a sharp feature on a rigid body, our method can guarantee that there will be always one node at the collision point to transit the material points from one side to the other. Thus, the rod can slide smoothly along the edge without any obvious artificial jittering even only using a small number of elements. Moreover, notice that the discretization is denser near the collision point, and it is exactly caused by our *r*-adaptive strategy for reallocating material points, which further helps to improve the smoothness near the contact region for rod sliding. In another example illustrated by Figure 10, the rods are falling and wrapping around the obstacles, where the discretization for rods and cylinders are nicely matched up as shown by the close-up figure.

A cracking whip Figure 9 illustrates one of the screenshots from a sequence of a simulated cracking whip. As for initialization, one end of the rod is given an instant impulse to revolve it. When the loop propagates forward, its size will shrink with a decreasing radius. By using our method, material points can automatically slide to the loop region, preserving its geometry smoothly and accurately. As a comparison, our simulated result is similar to the real-world experiment.

Knot sliding Starting from a shape with knots, our method can perfectly simulate the knot sliding motivated by dragging two ends of the rod. As shown in Figure 11, our method can push the material points to concentrate in the middle of the rod, where the knot loop is gradually shrinking with increasing curvature values. Finally, under

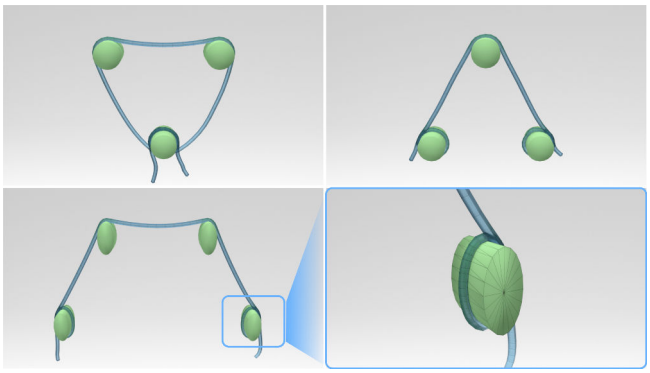


Figure 10: ***Rod wrapping****. In three different settings of rigid obstacles, the rod is falling under the gravity and is gradually wrapping around the obstacles. As the close-up figure shows on the bottom right corner, dynamic node insertion can adapt the reference mesh to tightly match up with the discretization of rigid bodies, producing an artifact-free contact simulation.*

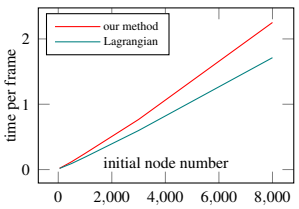
Figure	$\#V_{min}$	$\#V_{max}$	Collision	Resample	Force	Solve	Total
Fig.8	11	23	0.0988	0.0194	1.18	14.6	15.9
Fig.7	20	88	0.989	0.507	4.14	28.6	34.2
Fig.9	36	36	/	/	1.93	17.3	19.2
Fig.10(4 cylinders)	49	153	0.724	0.410	5.55	29.7	36.4
Fig.11(blue)	47	53	0.408	0.0162	2.94	21.0	24.4
Fig.11(green)	107	115	0.952	0.0384	5.98	36.5	43.5
Fig.11(red)	81	87	0.642	0.0489	3.92	29.2	33.8
Fig.11(yellow)	73	76	0.550	0.0455	3.75	26.1	30.4

Table 1: ***Timing statistics****. For examples depicted in the result section, we count the minimal and maximal number of nodes allocated in our rh-adaptive simulation, together with the averaged time cost (in milliseconds) spent in each step in Algorithm 1 per frame.*

the redistribution of the Eulerian points, a small but highly curved knot loop can be successfully reproduced which finalizes the knotting process. To clearly show the dynamics of material points in these cases, node insertion is solely based on the length tolerance to control a bounded number of samplings.

5.4. Timing

We roughly compare the performance of our method with a pure Lagrangian method in terms of average time cost (counted in second) per frame with varying numbers of initial sample points. As a comparison, the pure Lagrangian method follows [ST07] and is also implemented in our numerical framework with an implicit Euler integration. For fairness, we turn off the option for *h*-adaption in both methods. In each step, solving the linearized equation still dominates the computational cost. As shown in the inset, the time cost of our method is slightly larger than the pure Lagrangian version, which is quite reasonable since the extended Eulerian coordinates will introduce more DoFs, and thus the size of the system matrix is becoming larger. However, since the additional DoFs for Eulerian coordinates only occupy a small portion of the total size, it will not bring in too much overhead especially when the linear solver is well scaled. Our current implementation is not well optimized, and exploring multi-threading or using a more sophisticated linear solver will definitely help to greatly improve the run-time performance. Detailed statistics are provided in Table 1.



6. Limitations and Future Works

Though being successful to generate realistic simulation, we should note that we have made some approximation in our methods which might lead to inaccuracy in some cases. If the external impulse is violent, the reference mesh does have a chance to undergo intense motions. In such circumstances, our approximated EoM might cause some notable visual artifacts or severe energy dissipation. Besides, friction is not considered in our current implementation yet, which might fail to reproduce viscous dynamics like simulating interactions of rough materials, etc.

As for *r*-adaption, our prior to redistribute the sample points is completely geometry-based. Although it often exhibits good consistency with intensity distribution of physical field, there could be considerable discrepancy when rod material is becoming quite inhomogeneous. Therefore, how to model a more general, physics oriented prior to move those material coordinates, or even take account of posterior knowledge if groundtruth validation can be known in advance to improve the accuracy of *r*-adaptive method will be definitely an interesting and important future work.

Since having a stable and efficient *h*-adaption is much more challenging, our method could share similar shortcomings as previous works. For example, tiny segments generated in the simulation are simply merged for numerical conditioning, but this could be still problematic especially when multiple stacked rod layers are sliding towards each other, e.g. in yarn-level cloth simulations. While such scenarios are not common in our experiments, integrating the techniques of [SBRBO20] into our method is definitely an elegant alternative to resolve the infinitely large stiffness.

Finally, incorporating Eulerian coordinates numerically leads to more nonlinearity, degeneracy, poor conditioning, and thus great challenges in both computational efficiency and robustness. We believe that there should be better numerical methods to solve Equation (15) or its simplified version. For instance, the augmented Lagrangian method is probably worth trying. The inspiring idea of statics-dynamics separation in [SBRBO20] is also an interesting direction to explore.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments and suggestions. This work was supported by National Key R&D Program of China (No. 2017YFB1002703), JST CREST, JPMJCR17A1 (HCI for Machine Learning), Japan.

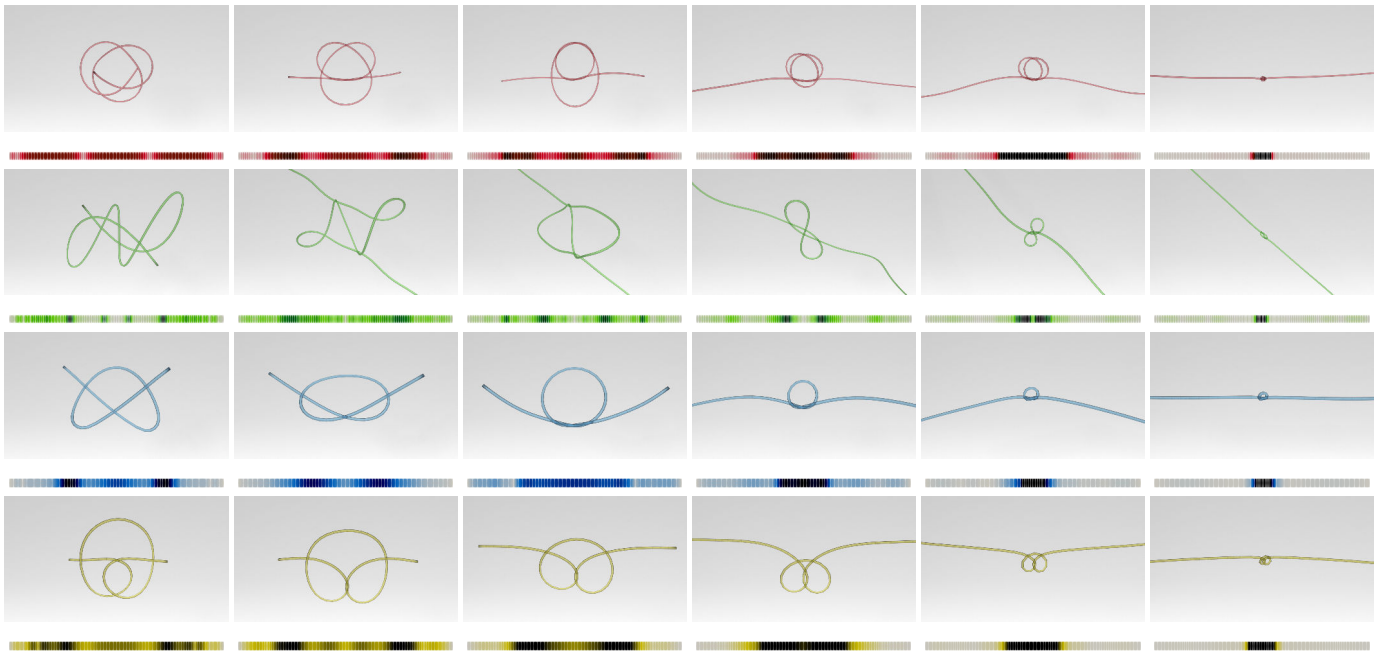


Figure 11: ***Knot sliding***. *By initializing the rod shape via different knot curves (red, green, blue and yellow color for each) and dragging two ends of the rod, our method can simulate the knotting process involving abundant bend, twist and contacts. Notice that when approaching the last stage of simulation, due to our curvature driven r-adaption, the material points are pushed to concentrate in the middle of the rods, where a small but sharply curved knot loop is successfully reproduced. The reference meshes are illustrated by the colored bar below each figure, whose color indicates the local curvature magnitude with deeper color for a larger value.*

References

[ApS19] APS M.: *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. URL: http://docs.mosek.com/9.0/toolbox/index.html. 8

[ART03] ANDERSEN E. D., ROOS C., TERLAKY T.: On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming 95*, 2 (2003), 249–277. 8

[BAC*06] BERTAILS F., AUDOLY B., CANI M.-P., QUERLEUX B., LEROY F., LÉVÊQUE J.-L.: Super-helices for predicting the dynamics of natural hair. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 1180–1187. 2

[BAV*10] BERGOU M., AUDOLY B., VOUGA E., WARDETZKY M., GRINSPUN E.: Discrete viscous threads. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 1–10. 2

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 594–603. 8

[BHR09] BUDD C. J., HUANG W., RUSSELL R. D.: Adaptivity with moving grids. *Acta Numerica* (2009), 1–131. 2, 3

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*. 2008, pp. 1–12. 2

[CC07] COSSERAT E., COSSERAT F.: *Sur la statique de la ligne deformable*. Gauthier-Villars, 1907. 2

[CC09] COSSERAT E., COSSERAT F.: *Théorie des corps déformables*. A. Hermann et fils, 1909. 2

[CLMMO14] CIRIO G., LOPEZ-MORENO J., MIRAUT D., OTADUY M. A.: Yarn-level simulation of woven cloth. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 1–11. 2, 3, 5

[CLMO16] CIRIO G., LOPEZ-MORENO J., OTADUY M. A.: Yarn-level cloth simulation with sliding persistent contacts. *IEEE Transactions on Visualization and Computer Graphics 23*, 2 (2016), 1152–1162. 3

[FLLP13] FAN Y., LITVEN J., LEVIN D. I., PAI D. K.: Eulerian-on-lagrangian simulation. *ACM Transactions on Graphics (TOG) 32*, 3 (2013), 1–9. 2, 3, 4, 5

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 8

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: a simple framework for adaptive simulation. *ACM transactions on graphics (TOG) 21*, 3 (2002), 281–290. 3

[HR10] HUANG W., RUSSELL R. D.: *Adaptive moving mesh methods*, vol. 174. Springer Science & Business Media, 2010. 2, 3

[Kir59] KIRCHHOFF G.: Ueber das gleichgewicht und die bewegung eines unendlich dünnen elastischen stabes. *Journal für die reine und angewandte Mathematik 1859*, 56 (1859), 285–313. 2

[KS16] KUGELSTADT T., SCHÖMER E.: Position and orientation based cosserat rods. In *Symposium on Computer Animation* (2016), pp. 169–178. 2

[LSNP13] LI D., SUEDA S., NEOG D. R., PAI D. K.: Thin skin elastodynamics. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–10. 3

[LWL*09] LIU Y., WANG W., LÉVY B., SUN F., YAN D.-M., LU L., YANG C.: On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics (ToG) 28*, 4 (2009), 1–17. 5

[MWN*17] MANTEAUX P.-L., WOJTAN C., NARAIN R., REDON S., FAURE F., CANI M.-P.: Adaptive physically based models in computer graphics. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 312–337. 3

[NPO13] NARAIN R., PFAFF T., O'BRIEN J. F.: Folding and crumpling adaptive sheets. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–8. 3, 7

[NSO12] NARAIN R., SAMII A., O'BRIEN J. F.: Adaptive anisotropic remeshing for cloth simulation. *ACM transactions on graphics (TOG) 31*, 6 (2012), 1–10. 3

[O'R08] O'REILLY O. M.: *Intermediate dynamics for engineers: a unified treatment of Newton-Euler and Lagrangian mechanics*. Cambridge University Press Cambridge, 2008. 2

[Pai02] PAI D. K.: Strands: Interactive simulation of thin solids using cosserat models. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 347–352. 2

[SBRBO20] SÁNCHEZ-BANDERAS R. M., RODRÍGUEZ A., BARREIRO H., OTADUY M. A.: Robust eulerian-on-lagrangian rods, 2020. URL: http://gmrv.es/Publications/2020/SRBO20. 3, 6, 10

[SJLP11] SUEDA S., JONES G. L., LEVIN D. I., PAI D. K.: Large-scale dynamic simulation of highly constrained strands. In *ACM SIGGRAPH 2011 papers*. 2011, pp. 1–10. 2, 3, 6

[SLNB10] SERVIN M., LACOURSIERE C., NORDFELTH F., BODIN K.: Hybrid, multiresolution wires with massless frictional contacts. *IEEE transactions on visualization and computer graphics 17*, 7 (2010), 970–982. 3

[SMSH18] SOLER C., MARTIN T., SORKINE-HORNUNG O.: Cosserat rods with projective dynamics. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 137–147. 2

[ST07] SPILLMANN J., TESCHNER M.: Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), pp. 63–72. 2, 4, 8, 10, 12

[ST08] SPILLMANN J., TESCHNER M.: An adaptive contact model for the robust simulation of knots. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 497–506. 2, 3

[USS14] UMETANI N., SCHMIDT R., STAM J.: Position-based elastic rods. In *ACM SIGGRAPH 2014 Talks*. 2014, pp. 1–1. 2

[WPLS18] WEIDNER N. J., PIDDINGTON K., LEVIN D. I., SUEDA S.: Eulerian-on-lagrangian cloth simulation. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–11. 2, 3, 7, 8

Appendix A: Discrete evaluation of energies

In this section, we will derive a discrete formula to calculate the energies used for the Cosserat model. The derivation is exactly following [ST07], and we just intend to provide a quick reference here for implementation purposes.

The continuous integral can be turned to summation over each element by using piecewise linear basis functions and one Gaussian quadrature. For the stretching potential V_s (Equation (5)), its discrete calculation per segment $\mathbf{e}_i$ can be written as

$$V_{s,i} = \frac{1}{2} k_s (u_{i+1} - u_i) \left(\frac{||\mathbf{x}_{i+1} - \mathbf{x}_i||}{u_{i+1} - u_i} - 1 \right)^2.$$

As for rotational kinetic energy (see Equation (6)), it is related to the angular velocity $\boldsymbol{\omega}$. With quaternion representation, each component of angular velocity ω_k along different directions is computed as

$$\omega_k = \frac{2}{||\mathbf{q}||^2} (\mathbf{B}_k \mathbf{q})^T \dot{\mathbf{q}},$$

where $\mathbf{B}_k \in \mathbb{R}^{4\times 4}$ is a constant skew-symmetric matrix:

$$\mathbf{B}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_2 = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \mathbf{B}_3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix},$$

and the temporal derivate can be later discretized via finite difference method for time integration. Between two adjacent line segments, both material frame and angular velocity are regarded as piecewise linear functions, and thus the rotational kinetic energy per segment is calculated as:

$$T_{r,i} = \frac{1}{8} (u_i - u_{i+1}) \sum_{k=1}^{3} \mathbf{J}_{kk} \left((\mathbf{B}_k (\mathbf{q}_i + \mathbf{q}_{i+1}))^T (\dot{\mathbf{q}}_i + \dot{\mathbf{q}}_{i+1}) \right)^2, \quad (18)$$

where three entries of diagonal inertia tensor is $\mathbf{J}_{11} = \mathbf{J}_{22} = \rho\pi r^2/4$ and $\mathbf{J}_{33} = \rho\pi r^2/2$ respectively. For twisting and bending, each component of frame strain ε_k is computed similarly, with the temporal derivative of $\mathbf{q}$ being replaced by its spatial derivative, namely

$$\varepsilon_k = \frac{2}{||\mathbf{q}||^2} (\mathbf{B}_k \mathbf{q})^T \nabla_u \mathbf{q}.$$

Having this formula, the discrete bending and twisting potential can be computed via

$$V_{b,i} = \frac{1}{2} (u_{i+1} - u_i) \sum_{k=1}^{3} \mathbf{K}_{kk} \left(\frac{2}{||\mathbf{q}_i||^2} (\mathbf{B}_k (\mathbf{q}_i + \mathbf{q}_{i+1}))^T \frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{l_i} - \bar{\varepsilon}_k \right)^2,$$

where $l_i = (u_{i+2} - u_i)/2$. The diagonal stiffness $\mathbf{K}$ has entries of $\mathbf{K}_{11} = \mathbf{K}_{22} = E\pi r^2/4$ and $\mathbf{K}_{33} = G\pi r^2/2$, where E and G are stiffness modulus which encodes the bending and torsional resistance respectively.

Appendix B: Equation of Motion for q

Starting from the kinetic energy T_r (Equation (6)) and potential energies V_b (Equation (7)), the equation of motion for material frame $\mathbf{q}$ can be similarly derived based on the Euler-Lagrange equation as

$$\text{EoM}(\mathbf{q}): \quad \dot{\mathbf{M}}_q(\mathbf{q})\dot{\mathbf{q}} + \mathbf{M}_q(\mathbf{q})\ddot{\mathbf{q}} - \frac{\partial T_r}{\partial \mathbf{q}} + \frac{\partial V_b}{\partial \mathbf{q}} = \tau, \quad (19)$$

where τ is the external twisting torque and $\mathbf{M}_q(\mathbf{q})$ is the mass for material frames. Given the discrete computation of T_r by Equation (18), the mass matrix is assembled as $\mathbf{M}_{q,i} = [1,1;1,1] \otimes \mathbf{m}_{q,i}$, where the sub-block is written as

$$\mathbf{m}_{q,i} = \frac{1}{8} (u_i - u_{i+1}) \sum_{k=1}^{3} \mathbf{J}_{kk} \left(\mathbf{B}_k (\mathbf{q}_i + \mathbf{q}_{i+1}) \right) \left(\mathbf{B}_k (\mathbf{q}_i + \mathbf{q}_{i+1}) \right)^T. \quad (20)$$

For time integration, we adopt implicit Euler same as before and drop some subtle terms for simplicity, and finally get

$$\mathbf{M}_q(\mathbf{q}^n) \frac{\mathbf{q}^{n+1} - \mathbf{q}^n - h\dot{\mathbf{q}}^n}{h^2} + \frac{\partial V_b}{\partial \mathbf{q}} (\mathbf{q}^{n+1}) = \tau. \quad (21)$$

This approximated EoM for $\mathbf{q}$ can be treated in the same way in the velocity solving process, as demonstrated in Section 5.2.