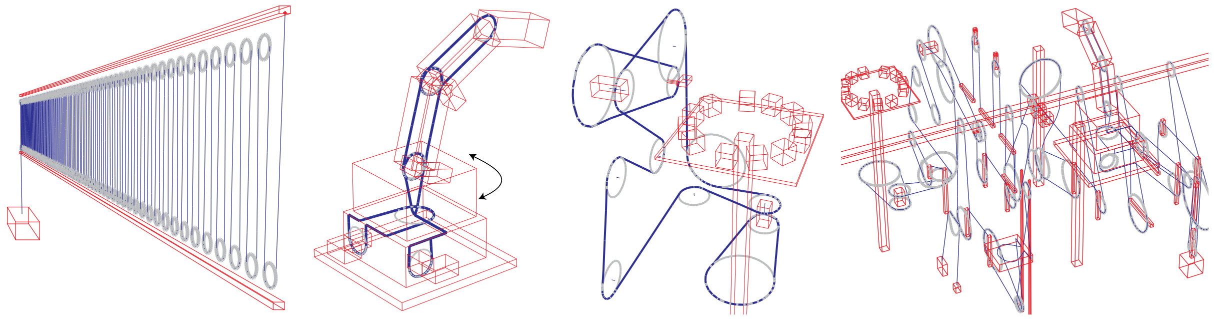
Large-Scale Dynamic Simulation of Highly Constrained Strands

Shinjiro Sueda Garrett L. Jones David I. W. Levin Dinesh K. Pai

Sensorimotor Systems Laboratory, University of British Columbia*



(a) *Compound pulley* **(b)** *Robotic arm* **(c)** *Chain gears* **(d)** *Line shaft*

Figure 1: *Various examples that demonstrate the scalability and robustness of our approach. (a) Compound pulley: our framework allows us to simulate a compound pulley system composed of up to 1024 pulleys connected by a single strand. (b) Robot arm: the cable for controlling the proximal arm is routed through a small aperture so that the arm can be controlled even when the robot body is rotated. (c) A cabled system with two 2:1 chain gear drives. (d) Large-scale simulation–subcomponents connected by an overhead line shaft.*

Abstract

A significant challenge in applications of computer animation is the simulation of ropes, cables, and other highly constrained strand-like physical curves. Such scenarios occur frequently, for instance, when a strand wraps around rigid bodies or passes through narrow sheaths. Purely Lagrangian methods designed for less constrained applications such as hair simulation suffer from difficulties in these important cases. To overcome this, we introduce a new framework that combines Lagrangian and Eulerian approaches. The two key contributions are the *reduced node*, whose degrees of freedom precisely match the constraint, and the *Eulerian node*, which allows constraint handling that is independent of the initial discretization of the strand. The resulting system generates robust, efficient, and accurate simulations of massively constrained systems of rigid bodies and strands.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Combined

Keywords: physically-based simulation, constrained strands, Lagrangian mechanics, elastic rods, thin solids

*{sueda,glj3,dilevin,pai}@cs.ubc.ca

1 Introduction

Many applications of computer graphics require the simulation of ropes, chains, belts, cables, tendons, hair, and other thin, curve-like physical objects. Using the terminology of Pai [2002] these objects are called *strands* to indicate that these are not just space curves but also have mass, elasticity, and other physical properties that influence their dynamics. During the last decade many efficient methods have been proposed for spatial discretization of strands for efficient dynamic simulation. See §1.2 for a brief review. These methods perform well when the strands are relatively unconstrained (e.g., hair strands fixed at one end and free at the other).

However, many important applications in engineering and biomechanics require strands to be highly constrained. Indeed, a major reason for using strand-like structures in engineering is that they are strong enough to transmit force axially, but can be wrapped around pulley-like structures or passed through holes (e.g., grommets) to constrain their movement. Robust, large-scale simulation of highly constrained strands, such as the 1024 pulleys in series (Figs. 1a, 8) or the line shaft scene (Fig. 1d), is difficult using previous methods. If the strand does not have enough degrees of freedom (DoF), interactions between the discretization of the strand and constraints on the strand's path can result in unexpected locking and other unintuitive behaviors. With our approach for handling constraints, the coupled dynamics of a wire inside a sheath can also be implemented, as shown in Fig. 3.

To understand these difficulties, consider a simple example shown in Fig. 2a[I]. A string passes through a long frictionless tube in a rigid body, like a bead on a necklace. The string is fixed to the world at its ends, and the rigid body is free to slide along the string. Notice that the string has to bend sharply as it exits the tube, if the bending stiffness of the string is low relative to the mass of the body (a common occurrence). To simulate the string as a strand, we discretize its geometry using a finite number of *nodes* (or control points) whose positions determine the shape of the strand. Note that this does not mean that mass is lumped at the nodes (though some methods do resort to this): mass and energy can be integrated

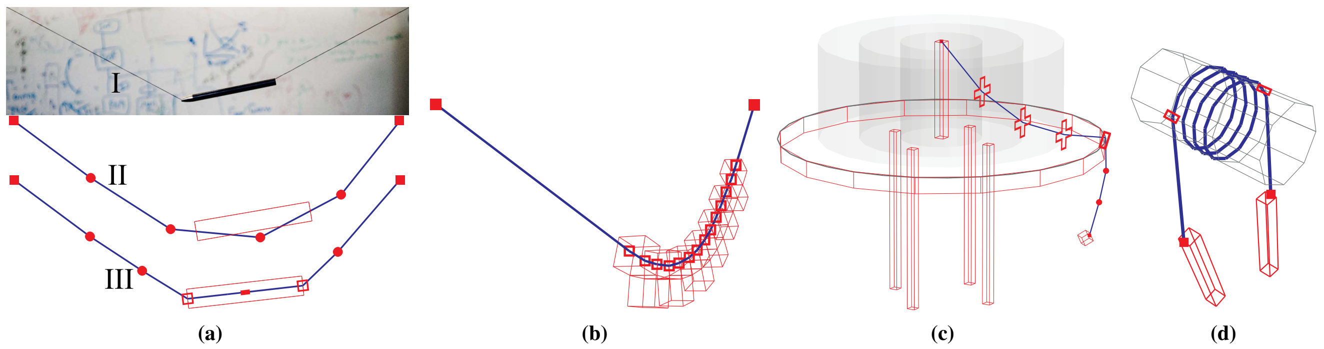


Figure 2: *(a) A simple example. (I) is a photograph of a string threaded through the barrel of a ball-point pen. Note the sharp kink in the string as it enters and exits the barrel. The typical fully Lagrangian method is shown in (II), and our proposed method in (III). E-nodes (open squares) are Eulerian nodes on the strand that are fixed to the tube, allowing it to bend sharply at the ends of the tube. A reduced node (solid dash) efficiently represents the motion of the strand inside the tube, exactly maintaining the constraint. These node types will be introduced later in §2.2. (b) A strand with only E-nodes can simulate this complex necklace without adding any Lagrangian degrees of freedom to the multi-rigid-body system. (c) A string hanging from the edge of a table. The E-node at the edge of the table is constrained on the circle, and the E-nodes between the pole and the table edge are constrained to lie on the transparent cylinders. The material of the strand flows freely through all these E-nodes. (d) Only four nodes are sufficient to simulate a strand wrapped 4 times around a shaft.*

along the length of the strand, as in the finite element method (e.g., [Qin and Terzopoulos 1996; Lenoir et al. 2004; Sueda et al. 2008]). We use linear interpolation between nodes for simplicity.

The typical way to handle constraints is to augment the dynamics with constraint equations. For example, Fig. 2a[II] shows sliding constraints [Lenoir et al. 2004] which allow the strand to slide along in the tangential direction at a particular point, but constrain the motion in the normal and binormal directions. This approach works well for simple constraints (like eyelets of shoelaces) but suffers from two problems. First, with sliding constraints, the strand cannot bend freely as it enters and exits the rigid body, causing rigid body to bounce as it slides along the strand, in the same way a chain cannot be pulled across the edge of a table smoothly. Using a higher order discretization, e.g., splines, does not solve the main problem, i.e., handling constraints; in fact it makes the problem more complex due to non-local coupling. (For instance, if a spline is constrained to be everywhere inextensible, it turns into a rigid body.) We can add more nodes to overcome some of these problems, but at the cost of either increasing the system size and decreasing the mass and length of each segment, which causes the system to become larger and more poorly conditioned. Second, we cannot directly constrain all the nodes inside the tube since this may over-constrain the system, causing locking. This problem is present even in this simple example and becomes worse in large-scale examples shown in Fig. 1.

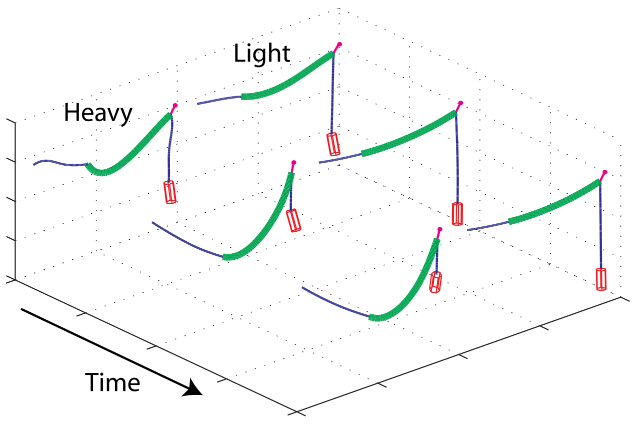


Figure 3: *A thin blue filament passes through a green casing. Two separate simulations are shown: heavy green casing (front row) and light green casing (back row).*

The underlying cause of all these problems is that the discretization of the strand is unrelated to the constraints. Specifically, the key problems are: (1) Many of the degrees of freedom (DoFs) are superfluous (e.g., inside the tube), and getting rid of them is non-trivial. (2) More subtly, the DoFs are not in the right places (e.g., not at the entrance of the tube). We address both these problems in a general framework for simulating constrained strands. Figure 2a[III] shows the example discretized using our method, which reproduces the desired behavior well with very few nodes.

The first problem is addressed by introducing *reduced nodes*, with precisely the number of degrees of freedom allowed by the constraints, leading to both simulation efficiency and exact constraint satisfaction. Figure 2a[III] shows a reduced node (with 1 DoF) inside the tube. Reduced coordinate descriptions are not new in computer graphics, especially for simulating rigid bodies constrained by joints (e.g., [Kry and Pai 2003; Lee and Terzopoulos 2008]). But our use of this general idea is new and it allows very general constraints (essentially any parametric curve or surface) to be imposed on deformable strands. Fig. 2d shows that a strand wrapped four times around a cylinder can be simulated by adding the degrees of freedom of only *two* 2 DoF nodes, taking the distribution of mass and strain energy in the curved segment fully into account. In our framework we can still include explicit constraints with Lagrange multipliers as usual (see §3.4-3.5), but the point is that many constraints can be efficiently and robustly handled using reduced nodes. We should point out that our reduced coordinates are subtly different from those used in reduced deformable models (e.g., modal models). Our reduced coordinates are used to eliminate degrees of freedom due to constraints, similar to those used in multibody dynamics, e.g., [Featherstone 1987]. They are also subtly different from simple joints in that any parameterized constraint of any dimension can be used for defining reduced nodes.

The second problem is addressed by introducing Eulerian nodes along with Lagrangian nodes. A Lagrangian node (which we will abbreviate as *L-node*) represents the motion of material points. This is the standard formulation used in solid mechanics. An Eulerian node, in contrast, represents motion of the strand through a point that is specified *in spatial coordinates*. The new type of node we introduce is very general: its position is specified with respect to *any* object whose position can be computed, e.g., a moving rigid body, as in [Servin et al. 2010], a curve or a surface fixed to a rigid body, or even another node. So it could be called something like Eulerian-on-Lagrangian, but for simplicity we will call it an *E-node*. The "pure" Eulerian case familiar from fluid mechanics is a special case, with the node fixed with respect to the world rather than to a moving object.

Why introduce the E-node? Because it allows us to locate nodes precisely at the constraints (which are specified with respect to objects), once and for all. In Fig. 2a[III] two E-nodes are located precisely at the ends of the tube. The strand can bend at a sharp angle there because the degrees of freedom of the node are always located at the right point on the strand, regardless of which material point happens to be at the node. The resulting discretization can capture the desired behavior with very few total degrees of freedom because they are well matched to the constraints. In fact we can take this to the extreme; in Fig. 2b, 13 rigid bodies are strung on a string, constrained only by E-nodes, adding *zero* Lagrangian DoFs to the rigid body system. But since we compute the dynamics of the strand using the Eulerian degrees of freedom at the E-nodes, we still get a rich set of dynamic behavior of the strand.

The E-node is a generalization of "via points" from [Delp et al. 2007], "pulley points" from [Kiss et al. 1999] and [García-Fernández et al. 2008], and "massless contact nodes" from [Servin and Lacoursiere 2007; Servin et al. 2010], all of which correspond to just the E0-node in Fig. 5a. Our formulation is a significant generalization which can mix and match the 8 node types shown in the figure, allowing us to combine reduced and maximal coordinates, as well as Eulerian and Lagrangian nodes.

1.1 Contributions

We describe a general simulation framework for simulating highly constrained systems of rigid bodies and strands, such as the complex machines shown in Fig. 1. We address the major difficulties faced by previous methods, by generalizing the standard Lagrangian formulation in two independent directions. First, nodes can be *reduced* in their degrees of freedom to precisely match the constraints. Second, we introduce a new Eulerian type of node (called *E-node*) into the standard Lagrangian formulation, which allows the degrees of freedom to be located with respect to constraints, rather than with respect to the material. In addition, our system is able to account for the continuous distribution of mass and energy along the length of the strand, rather than lumping these quantities at the nodes.

1.2 Related Work

The simulation of three-dimensional space curves has recently received considerable attention in graphics. Research has focused on the physical simulation of hair [Choe et al. 2005; Bertails et al. 2006; Selle et al. 2008; McAdams et al. 2009], cloth [Nocent and Remion 2001; Kaldor et al. 2010], threads [Kubiak et al. 2007; Bergou et al. 2010], needles and catheters [Lenoir et al. 2006; Chentanez et al. 2009], tendons, muscles and related biomechanical systems [Lenoir et al. 2004; Sueda et al. 2008], as well as cabled mechanisms [Servin and Lacoursiere 2007; García-Fernández et al. 2008]. Motivated by such critical application areas there has also been a focus on the development of general-purpose, physically-based curve models [Pai 2002; Hadap 2006; Spillmann and Teschner 2008; Bergou et al. 2008; Martin et al. 2010].

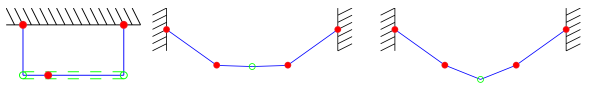

(a) *Tube* **(b)** *Bead*

Figure 4: *Toy examples to show the undesirable effects of mass lumping. (a) A rigid tube hanging on a wire. If the mass of the strand is integrated along the strand, then the tube stays in equilibrium no matter where the node is within the tube. If, however, the mass of the strand is lumped onto the node, then the node must be exactly in the middle of the tube. (b) A small bead with a massless contact node is inserted in the middle of the strand. If the mass of the strand is lumped onto the red nodes, then the strand in equilibrium does not bend at the middle node, since the mass of the strand near the middle node has been lumped away onto the red nodes. On the other hand, if the mass is integrated along the strand, then it takes on the expected catenary-like shape.*

Thin solids can be modeled using different parameterizations of the degrees of freedom, each with its advantages and disadvantages. Recently proposed parameterizations include Super-Helix models [Bertails et al. 2006; Bertails 2009], spline bases [Qin and Terzopoulos 1996; Remion et al. 1999; Nocent and Remion 2001; Lenoir et al. 2004; Coleman and Singh 2006; Theetten et al. 2008; Sueda et al. 2008; Kaldor et al. 2008], quaternion representations [Grégoire and Schömer 2007; Spillmann and Teschner 2009], elastic rod models [Pai 2002; Bergou et al. 2008], beam elements [Lenoir et al. 2006], rigid body links [Choe et al. 2005; Hadap 2006], and mass-spring systems [Selle et al. 2008; McAdams et al. 2009]. All of these methods are designed mainly for strands that are relatively unconstrained, and would require multiplier or penalty based methods in order to constrain the path of the strands. The constraints in our system, on the other hand, are built into the formulation of the nodes (§2.2) and segments (§2.3) of the strand, and this enables us to handle the constraints independently from the spatial discretization of the strand. This is a key advantage, since we are interested in the scalable simulation of strands for the transmission of force around a large number of constraints of various types.

All the above methods are essentially Lagrangian. In computational mechanics Arbitrary Lagrangian-Eulerian (ALE) methods have been proposed (see [Donea et al. 2004] for a review). Even though our method shares some of the motivation, there are important differences. Rather than move the Lagrangian nodes to track evolving phenomena, we use Eulerian nodes to represent constraint transitions efficiently, providing the user with a powerful and flexible modeling primitive. In Fig. 2c, for example, the material of the strand is allowed to pass through the three Eulerian nodes whose spatial coordinates are constrained to lie on the vertical cylinders.

Our work is most closely related to the work of Servin et al. [2007; 2010]. The main advantages of our method are in the full range of node and segment types supported, with no mass lumping. The massless contact nodes of Servin et al. has the advantage that frictional forces are easier to apply, and collision detection and adaptive resampling are already included in their implementation.

2 Strand Graph

Just as the dependencies of various transforms of a 3D scene can be described by a scene graph, the dependencies of the kinematics of

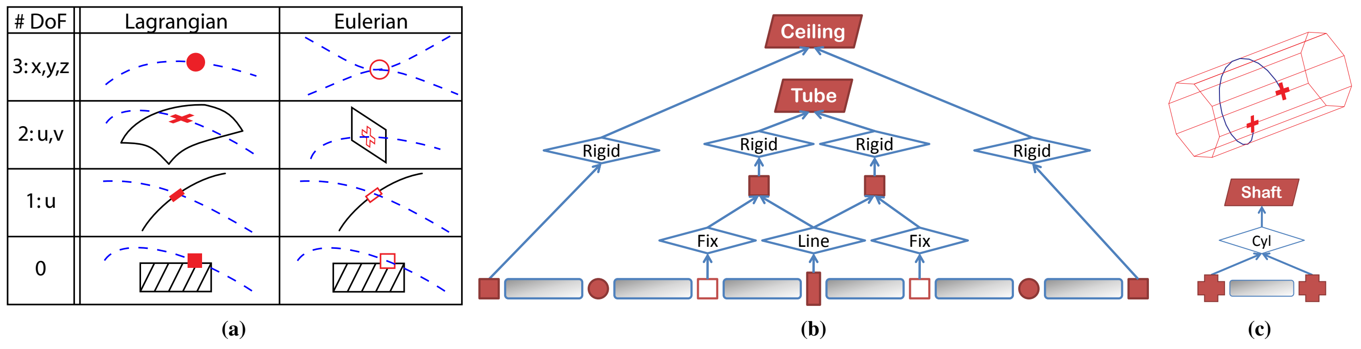


Figure 5: *(a) Table of node types. The first column shows the number of degrees of freedom. The last two columns show the icons used for various nodes and sketch example usage scenarios. In addition, rigid bodies (6-nodes) are indicated by a parallelogram. (b) Strand graph for Fig. 2a(III). Constraints between nodes are represented with diamonds, and strand segments are represented with horizontal rectangles. (One of the two L3-nodes to the left of the Tube rigid body is not shown in the graph.) (c) A cylinder strand and its graph. With our approach, only two L2-nodes are required to represent the helical strand.*

nodes and other data structures representing a system of constrained strands and rigid bodies can be described by a "strand graph". The graph for the scene in Fig. 2a is shown in Fig. 5b. We now describe each of the elements of a strand graph.

2.1 Constraints

We start with constraints since our main goal is to deal efficiently with them. The key is to represent the majority of constraints as *parametric* curves and surfaces attached to rigid bodies and nodes. More formally, a constraint is any function $\boldsymbol{x}(q)$ that computes the 3D position from the generalized coordinates q of the nodes and the rigid bodies. We will see below that this allows efficient implementation of reduced nodes. Explicit constraints can be dealt with using Lagrange multipliers as usual (e.g., §3.4).

2.2 Nodes

In our framework, nodes are not mass-points of the strand, but rather, they represent the strand's degrees of freedom (DoF). Fig. 5a shows the different types of nodes. Each node can have a different number of "spatial," or Lagrangian DoFs, denoted by $\widetilde{x}$. The tilde ($\sim$) is used as a reminder that this DoF may be a reduced DoF, e.g., on a curve or a surface, and need not be in $\mathbb{R}^3$. We will call a node with 3 DoF, a *3-node*, etc., and if that node is an L-node, then we will call it an *L3-node*. The spatial coordinates, $\widetilde{x}_i$, of each node is different depending on the type of the node. For example, computing the world position, $\boldsymbol{x}_i$, of a 3-node is simply $\boldsymbol{x}_i(\widetilde{x}_i) = \widetilde{x}_i$, and for a node with less than the full, 3 DoF, we need the mapping between its generalized spatial coordinates and the world position. Each 2-, 1-, and 0-node (or "reduced" node) is reduced to a *parametric* constraint of dimension ≤ 2, and the DoFs, $\widetilde{x}_i$, of the node are the variables of the parametric constraint function.

Starting at a leaf node, we can traverse the graph vertically to compute the kinematics of a node. For example, the two end nodes in Fig. 5b are 0-nodes, with "rigid" constraints with respect to the ceiling rigid body. To compute the node's position, we start with the spatial coordinates of the node, which is $\widetilde{x}_i = \emptyset$ in this case, and apply it to the rigid constraint. We stop once we hit the rigid body, (which, in this context, can be thought of as a 6-node), since there are no more outgoing edges to follow. There are two other types of constraint functions in Fig. 5b. The "Fixed" constraint function simply equates the positions of the child and parent nodes. The "Line" constraint enforces the child node to be on a line between two parent nodes. The "Cylinder" constraint in Fig. 5c takes the child node's 2D state $\widetilde{x}_i = [u, v]^T$ and returns its world position by looking up the parent 6-node.

The strand graph gives us a convenient map of information flow between different objects in the scene. For example, we can easily see which node is affected by which other node, and this information can be used to determine which parts of the graph need to be revisited if a part of the scene is modified. Also, the map allows us to readily determine the total DoF of a node. Starting with a leaf node, we can traverse upward and concatenate the DoF at each parent node, until we reach the root.

A node can be Lagrangian or Eulerian, from the point of the view of the strand. An E-node allows the material of the strand to pass through it, whereas an L-node is tied to a specific material point on the strand. Along with the spatial DoFs, $\widetilde{x}_i$, each E-node also has a "material," or Eulerian DoF, s_i, which is the material coordinate of the strand at that node. (For an L-node, s_i is fixed.) Any parameterization of the strand can be used for s, but we use arc-length parameterization for simplicity. s is defined in the strand's reference configuration; $s = 0$ at the first node, and $s = L$ at the last node, where L is the total length of the strand at rest. For visualization purposes, s can also be treated as a texture coordinate of the strand, since it is defined in reference space. For the i^{th} node, we concatenate its material coordinate to the spatial coordinates of the node to get the total DoF of the node: $q_i = (\widetilde{x}_i, s_i)^T$.

2.3 Segments

The mass and energy of a strand are stored in its segments, rather than at its nodes as in a mass-lumped model. To compute the equations of motion, we traverse the segments, which are laid out horizontally in the strand graph. The geometric path of the segment between two nodes is interpolated in the *shared constraint* space of the two nodes. For example, the two strand nodes in Fig. 5c share the same cylinder constraint—both nodes are 2-nodes, and lie on the same cylinder, parameterized by (u, v). In such a case, the geometric path of the segment between the nodes is interpolated in the parameter space of the cylinder, giving us a helix in world space. In our current implementation, the u and v coordinates are independently interpolated linearly, but our approach can be extended to higher order schemes, such as Hermite interpolation, if desired.

In the strand graph, starting from the two nodes of a segment, we traverse vertically until a shared constraint is found, or the root is

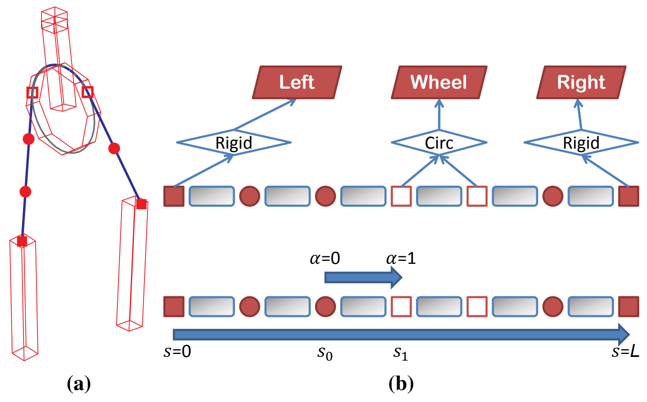


Figure 6: *A pulley strand (a) and its associated graph (b). The material coordinate, s, is defined for the whole strand, and goes from 0 to L, where L is the length of the strand at rest. α is defined in each segment, and goes from* 0 *to* 1.

reached. If the root is reached first, then the two nodes do not share a constraint, so the constraint space is the world space: the two segments are simply connected by a straight line in $\mathbb{R}^3$. However, if a shared constraint is found, then the segment's geometric path is interpolated in that constraint's space.

3 Equations of Motion

To derive the equations of motion, we first construct the kinetic energy, T, and potential energy, V, of the strand, and then form the equations of motion from these quantities:

$$M\ddot{q} = -\frac{\partial V}{\partial q} + \left(\frac{\partial T}{\partial q} - \dot{M}\dot{q}\right). \tag{1}$$

The terms in brackets are the quadratic velocity vectors that result from the position dependence of the kinetic energy. In the rest of this section, we will derive the equations of motion for a general strand involving various types of nodes and segments, interspersed with concrete examples for clarity.

Let $q_0 = (\tilde{x}_0, s_0)^T$ and $q_1 = (\tilde{x}_1, s_1)^T$ denote the DoFs of the left and right nodes of a segment. For simplicity, we will assume that strain is constant in each segment. Given a *material* coordinate $s \in [s_0, s_1]$, we can compute the interpolated *spatial* coordinates of the material point at s: $\tilde{x}(s) = (1-\alpha)\tilde{x}_0 + \alpha\tilde{x}_1$, where $\alpha = (s-s_0)/(s_1-s_0)$. See Fig. 6b for an illustration. From this, we can compute the material position in world space using the constraint function, $\boldsymbol{x}(\tilde{x})$, and the material velocity is then computed using the chain rule:

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{x}}{\partial \tilde{x}}\left(\frac{\partial \tilde{x}}{\partial q_0}\dot{q}_0 + \frac{\partial \tilde{x}}{\partial q_1}\dot{q}_1\right). \tag{2}$$

After some rearranging, we have

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{x}}{\partial \tilde{x}}\left((1-\alpha)\dot{\tilde{x}}_0 + \alpha\dot{\tilde{x}}_1 - \frac{\Delta\tilde{x}}{\Delta s}\left((1-\alpha)\dot{s}_0 + \alpha\dot{s}_1\right)\right), \tag{3}$$

where $\Delta\tilde{x} = \tilde{x}_1 - \tilde{x}_0$ and $\Delta s = s_1 - s_0$. For concreteness, if we assume that the two nodes are connected by a line, then the interpolation is in world space, and we have

$$\dot{\boldsymbol{x}} = (1-\alpha)\dot{\boldsymbol{x}}_0 + \alpha\dot{\boldsymbol{x}}_1 - \frac{\Delta\boldsymbol{x}}{\Delta s}\left((1-\alpha)\dot{s}_0 + \alpha\dot{s}_1\right). \tag{4}$$

And if we assume instead that the two nodes share the same "circle" constraint space parameterized by the angle, $\tilde{x}_i = \tilde{\theta}_i$, about the circle, we have

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{x}}{\partial \tilde{\theta}}\left((1-\alpha)\dot{\tilde{\theta}}_0 + \alpha\dot{\tilde{\theta}}_1 - \frac{\Delta\tilde{\theta}}{\Delta s}\left((1-\alpha)\dot{s}_0 + \alpha\dot{s}_1\right)\right). \tag{5}$$

In both cases, the first two terms can be interpreted as the contribution to the material velocity from the "Lagrangian" motion of the nodes, and the last two terms can be interpreted as the "Eulerian" correction term due to the flow of material through the nodes.

3.1 Kinetic Energy

The kinetic energy of the segment is computed by integrating the infinitesimal contributions from s_0 to s_1

$$T = \frac{1}{2}\int_{s_0}^{s_1} \rho\dot{\boldsymbol{x}}^T\dot{\boldsymbol{x}}ds, \tag{6}$$

where ρ is the linear density of the strand. By integrating out s, we obtain

$$T = \frac{1}{2}\begin{pmatrix}\dot{\tilde{x}}_0^T & \dot{\tilde{x}}_1^T & \dot{s}_0 & \dot{s}_1\end{pmatrix} M \begin{pmatrix}\dot{\tilde{x}}_0\\ \dot{\tilde{x}}_1\\ \dot{s}_0\\ \dot{s}_1\end{pmatrix}, \tag{7}$$

where M is the generalized mass matrix of the segment.

$$M = \frac{\rho}{6}\begin{pmatrix}2m_{xx} & m_{xx} & -2m_{xs} & -m_{xs}\\ \cdot & 2m_{xx} & -m_{xs} & -2m_{xs}\\ \cdot & \cdot & 2m_{ss} & m_{ss}\\ \cdot & \cdot & \cdot & 2m_{ss}\end{pmatrix}, \tag{8}$$

where the dots indicate symmetry. $m_{xx} = \Delta s\tilde{X}$, $m_{xs} = \tilde{X}\Delta\tilde{x}$, and $m_{ss} = \frac{\Delta\tilde{x}^T\Delta\tilde{x}}{\Delta s}\tilde{X}$, where $\tilde{X} = (\frac{\partial\boldsymbol{x}}{\partial\tilde{x}})^T(\frac{\partial\boldsymbol{x}}{\partial\tilde{x}})$. For the "Line" constraint, $\tilde{X} = I$, and for the "Circle" constraint, $\tilde{X} = a^2$ where a is the radius.

3.2 Gravity

To derive the force of gravity, we start with the gravitational potential of a segment. Letting $\boldsymbol{g}$ denote the gravity vector,

$$V = \rho\boldsymbol{g}^T\int_{s_0}^{s_1}\boldsymbol{x}ds. \tag{9}$$

The forces are derived by taking the derivative with respect to the DoFs of the nodes. For example, it can be computed easily for the line segment constraint:

$$\begin{aligned} V &= \frac{1}{2}\rho\boldsymbol{g}^T\Delta s(\boldsymbol{x}_0 + \boldsymbol{x}_1)\\ \frac{\partial V}{\partial q} &= \begin{pmatrix}\frac{\partial V}{\partial \boldsymbol{x}_0} & \frac{\partial V}{\partial \boldsymbol{x}_1} & \frac{\partial V}{\partial s_0} & \frac{\partial V}{\partial s_1}\end{pmatrix}\\ &= \frac{1}{2}\rho\boldsymbol{g}^T\begin{pmatrix}\Delta s & \Delta s & -(\boldsymbol{x}_0+\boldsymbol{x}_1) & (\boldsymbol{x}_0+\boldsymbol{x}_1)\end{pmatrix}. \end{aligned} \tag{10}$$

3.3 Bending Energy

Discrete bending forces are added between each segment, and continuous bending forces are added to a segment if the segment is on a curved constraint. We extend the formulation of discrete bending energy by Bergou et al. [2008] to work with E-nodes and non-linear segments. First, we assume that for the purpose of computing the bending energy, the strand should resist bending at an E-node in exactly the same way as it would at an L-node. So in computing the bending energy, E-nodes and L-nodes can be treated in the same way. Second, because segments need not be linear, the bending angle must be defined not with the two edges incident on a node, but with the two local tangents at the node.

Continuous bending energy can be derived for any segment, as long as the two nodes of the segment share a common constraint function. For example, the curvature of a segment on a cylindrical constraint parameterized by $\widetilde{x}_i = (\widetilde{\theta}_i, \widetilde{v}_i)^T$ is $\kappa = a\Delta\widetilde{\theta}^2/l^2$, where a is the radius of the cylinder, $\Delta\widetilde{\theta} = \widetilde{\theta}_1 - \widetilde{\theta}_0$, $\Delta\widetilde{v} = \widetilde{v}_1 - \widetilde{v}_0$, and $l = \sqrt{a^2\Delta\widetilde{\theta}^2 + \Delta\widetilde{v}^2}$ is the length of the segment. Then we can define the bending potential as $V = (1/2)K_b\kappa^2\Delta s$, and take its derivative with respect to the DoFs of the segment's nodes. Due to the bending potential, the helical strand in Fig. 5c between two L2-nodes tries to straighten and align with the axis of the cylinder.

3.4 Inextensibility

Inextensibility is handled by adding Lagrange multiplier constraints. Because we use reference-space arc-length parameterization for s, the strain of a segment is simply

$$\varepsilon(q_0, q_1) = \frac{l(\widetilde{x}_0, \widetilde{x}_1)}{\Delta s} - 1, \tag{11}$$

where l is the length of the segment. The constraint to keep the strain constant is arrived at by rearranging the terms:

$$g = l^2 - \Delta s^2$$
$$\dot{g} = G\dot{q} = 2\begin{pmatrix} l\frac{\partial l}{\partial \widetilde{x}_0} & l\frac{\partial l}{\partial \widetilde{x}_1} & \Delta s & -\Delta s \end{pmatrix}\begin{pmatrix} \dot{\widetilde{x}}_0 \\ \dot{\widetilde{x}}_1 \\ \dot{s}_0 \\ \dot{s}_1 \end{pmatrix}. \tag{12}$$

For example, for the line segment constraint, $l = \|\Delta \boldsymbol{x}\|$, and for the circle segment constraints, $l = a\|\Delta\widetilde{\theta}\|$.

3.5 Infinite Friction

Infinite friction is useful for coupling the motion of a strand relative to its parent rigid body. A good example of this is a geared pulley driven by a chain. The relative velocity between the chain and the gear should be zero at all points in contact. Assuming that strands are inextensible, we can achieve this by applying a non-slip constraint at only *one* point between the strand and the parent rigid body. If there is no slip at the chosen point, then there will be no slip at any other point in contact.

The infinite friction constraint can be applied to a segment only if its two nodes share a common constraint. For example, let us consider the pulley strand in Fig. 6a, whose middle segment connects two nodes that share the same circle constraint. We constrain the relative motion of a material point with respect to $\boldsymbol{x}_p$, the corresponding position on the parent of the circle constraint. Any point along the segment can be picked for this constraint, and so we pick the midpoint, corresponding to $\alpha = 0.5$. We then project this relative motion onto the tangent vector of the segment, ($\boldsymbol{t} = \partial\boldsymbol{x}/\partial\widetilde{\theta}_i$ for the circle constraint), and linearize, assuming that the tangent is constant. Such an approximation is common in contact dynamics, where the contact normal between two bodies are assumed to be constant during a time step [Kaufman et al. 2008]. The positional constraint and its derivative are then

$$g = \boldsymbol{t}^T(\boldsymbol{x} - \boldsymbol{x}_p), \quad \dot{g} = \boldsymbol{t}^T(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_p). \tag{13}$$

For the circle segment constraint, this can be rearranged as

$$\dot{g} = \frac{a^2}{2}\begin{pmatrix} 1 & 1 & -\frac{\Delta\widetilde{\theta}}{\Delta s} & -\frac{\Delta\widetilde{\theta}}{\Delta s} \end{pmatrix}\begin{pmatrix} \dot{\widetilde{\theta}}_0 \\ \dot{\widetilde{\theta}}_1 \\ \dot{s}_0 \\ \dot{s}_1 \end{pmatrix}. \tag{14}$$

This implies that the flow of material due to the node's Lagrangian DoF must match the change due to the Eulerian DoF.

4 Time Integration

The strand framework is agnostic to the choice of integrator. Using the mass matrix M from §3.1, the force vector $f = -\partial V/\partial q$ from §3.2, and constraint matrix G and vector g from §3.4-3.5, we can use any integrator we choose to step the state of the system forward in time.

In our current implementation, we take the two step approach common in computer graphics. We solve for the constrained velocities, and then update the positions using these new velocities. Assuming we know the active constraints at the current time step, we obtain a linear Karush-Kuhn-Tucker (KKT) system [Boyd and Vandenberghe 2004]:

$$\begin{pmatrix} M & G^T \\ G & 0 \end{pmatrix}\begin{pmatrix} \dot{q} \\ \lambda \end{pmatrix} = \begin{pmatrix} M\dot{q}_0 + hf \\ 0 \end{pmatrix}, \tag{15}$$

where $\dot{q}_0$ is the velocity at the previous time step and λ is the vector of Lagrange multipliers for the constraints. The position update is performed trivially for all the nodes: $q = q_0 + h\dot{q}$, where h is the time step size. For rigid bodies, we use the Rodrigues' formula [Murray et al. 1994].

Because the constraints are solved at the velocity level, the system may drift away from the constraint manifold over time. To fight this drift, we add a post-stabilization step [Cline and Pai 2003] after taking a position step.

$$\begin{pmatrix} M & G^T \\ G & 0 \end{pmatrix}\begin{pmatrix} \Delta q \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ -g \end{pmatrix}. \tag{16}$$

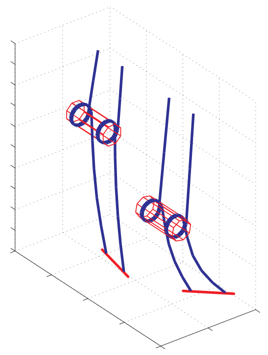
The generalized positions are updated similarly as before: $q = q_0 + \Delta q$.

4.1 Resampling

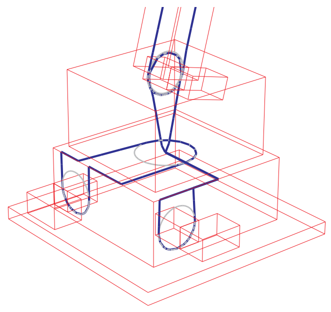
Remeshing and resampling are commonly used techniques in solid mechanics simulators, to prevent disproportionately small or thin elements from appearing and causing numerical difficulties. Because strands are 1D, remeshing is unnecessary. We just need to resample along the strand so that small elements are eliminated. There are a number of resampling schemes that have been proposed, such as optimizing for the minimum jump in bending energy [Spillmann and Teschner 2008], or resampling based on the tension in the strand [Servin et al. 2010]. Our resampling scheme is purely position based; we simply make sure that small elements do not arise, by adjusting the positions of the L-nodes in the system. More sophisticated schemes can be added if needed.

In addition, we resample the E-nodes on constraint curves and surfaces. This ensures that, at the point of contact, the strand is always tangential to the constraining curve or surface. This can be done efficiently using the Newton-Raphson method.

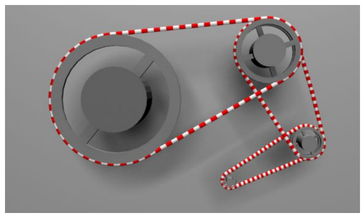
Although resampling is not a major contribution of this paper, we make two observations about our implementation. First, if the mass



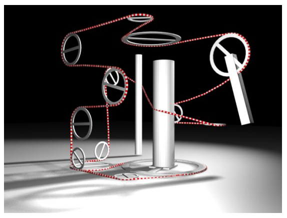
(a) *Cable elevator*



(b) *Robotic arm*



(c) *Chain drive*



(d) *Routing around columns*

Figure 7: *(a) Two frames from a simulation of a cylinder with infinite friction, which travels up and down the cables when a torque is applied to it. (b) The cable responsible for moving the proximal joint of the robotic arm goes through a small aperture in the base of the robot. This allows the cable to control the arm even when the body of the robot rotates. (c) Three 2:1 gear ratios resulting in gear ratio of 8:1. (d) The cable routed around the two vertical columns dynamically reacts to the tension and slides on the columns.*

of the strand is lumped onto the nodes, then there will be a jump in the kinetic energy whenever a strand is resampled. If, on the other hand, the mass is integrated along the strand as in our system, then changing the position of the nodes has negligible effect on the kinetic energy. Second, although resampling can cause constraint drift, the post-stabilization step fixes the drift by solving for the closest valid configuration using the kinetic metric. Thus, post-stabilization kills two birds with one stone: error caused by the numerical integrator, and the error caused by resampling.

5 Results

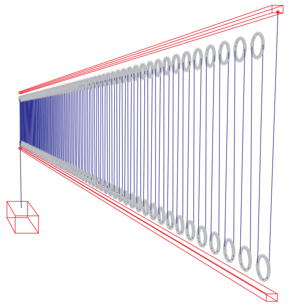
We implemented our system in Matlab and C++, and ran the simulations on a PC with an Intel Core i7 860 processor @ 2.8Ghz and 6GB of RAM. The code is single-threaded, and uses csparse [Davis 2006] for solving the sparse linear system. In our examples, the physical dimensions of the rigid bodies range from about 3 cm to 20 cm, and the strand radius from 1 mm to 3 mm.

Cable Climbing The example in Fig. 7a demonstrates a simple application of a parametric constraint defined on a dynamically moving object. Two cables are wrapped around a cylinder with infinite friction. A torque is applied to the cylinder with a simple feedback controller to make the cylinder go up and down the cables.

Robotic Arm The robotic arm in Fig. 1b and Fig. 7b shows a mechanism by which torque can be transmitted across a rotating body, using a small aperture for routing the cable. The steering wheel to the left of the base controls the rotation of the whole robot, and the wheel to its right controls the proximal joint. Even when the body is rotated, the cable responsible for controlling the proximal arm does not change its length, because the cable goes through the center of rotation.

Table 1: *Compound pulley test: Pulleys are inserted between two horizontal beams. A single strand is passed between all the pulleys, attached at one end to a hanging counter weight (box) and at the other end to the fixed upper beam. The lower beam weighs 1000 kg.*

Pulleys	Theoretical mass (kg)	Relative error
2	500.00	2e-5
4	250.00	4e-5
8	125.00	7e-5
16	62.50	1e-4
32	31.25	2e-4
64	15.63	5e-4
128	7.81	9e-4
256	3.91	1e-3
512	1.95	3e-3
1024	0.98	6e-3



Chain Drive In this example (Fig. 7c), a system involving three 2:1 chain gear boxes are simulated, resulting in a gear ratio of 8:1 between the smallest and the largest gears. Midway through the simulation, the direction is reversed to show the subtle dynamic effects of the cables.

Routing around Columns The cable in Fig. 7d is routed around two columns. Depending on the tension, the cable slides up and down the columns in a dynamic fashion.

Large-scale Compound Pulleys A distinct feature of a compound pulley system is that the force generated is linearly proportional to the number of pulleys implemented. We demonstrate an example illustrating how our framework can reproduce this relationship. The top beam is fixed in space, and the lower beam and the counter weight are both suspended by a single strand passing through all the pulleys. The strand starts at one end of the top beam, and is routed around an alternating set of pulleys attached to the lower and upper beams. Each pulley is a separate rigid body with a revolute joint, and rotates due to infinite friction between the strand and the pulley. After being routed around n pulleys, the strand is fixed to the hanging counter weight.

The mass of the lower beam is set to 1000kg. The theoretical mass of the hanging weight required to keep the system in equilibrium is then $(1000/n)$ kg. With the simulator, the force required to keep the system in equilibrium is calculated on-line for a varying number of pulleys, with a simple feedback controller that applies a small force to the counter weight. A near-perfect linear relationship is reproduced through our framework, as shown in Table 1, for up to 1024 pulleys in series connected by a single cable. The second column shows the theoretical mass of the counter weight for the number of pulleys in series. The third column shows the force computed by the controller, in terms of relative error from the theoretical mass. Even for the 1024 pulley model (see Fig. 8), the relative error is only a fraction of a percent. In the accompanying video, we show that adding just $\pm 1\%$ of the theoretical mass quickly drives the system out of equilibrium, as desired.

Line Shaft The final example is a large-scale simulation of a line shaft, driven with a simple feedback controller (Fig. 1d). Mechani-

Figure 8: *1024 pulleys in series connected by a single strand. We are able to simulate the dynamic interaction between the strand and all of the pulley rigid bodies robustly. Zoom in to see all 1024 pulleys.*

Table 2: *Computational cost for simulating one second of the pulley scene. The top row shows the number of pulleys. (Some of the smaller scenes are omitted from this table.) The second row shows the percentage of time spent in the computation of the system matrices and vectors. The third and fourth rows show the amount of time spent in the system solve, in absolute and relative terms. The fifth row shows the sparsity of the KKT matrix. As the number of pulleys increases, the cost of solving the linear system starts to dominate.*

# Pulleys	64	128	256	512	1024	2048	4096
fill (s)	0.11	0.22	0.45	0.90	2.0	4.2	9.0
solve (s)	0.14	0.36	0.89	3.11	12.3	48.0	189.7
solve (%)	55	62	67	78	86	92	95
sparsity (%)	6.8	3.7	2.0	1.0	0.50	0.25	0.13

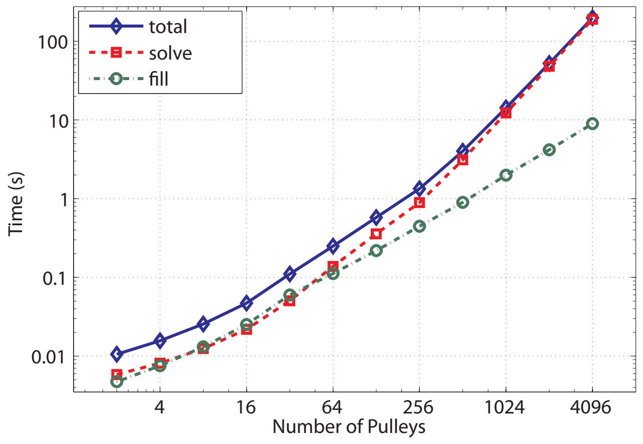


Figure 9: *A log-log plot of the computation time required to simulate one second of the pulley scene.*

cal power is transmitted from the overhead line shaft, using rocker-crank linkages to produce oscillatory motion, to various components, such as a compound pulley, a double-pulley, an elevator, a robotic arm, etc. Since there are no spurious bending and frictional effects in the system, the torque from the shaft is transmitted properly across numerous constraints. All of the sub-machines are simulated together and are dynamically coupled; if we forcibly jam any one of the components, the whole system will come to a grinding halt.

5.1 Performance Analysis

To show the scalability of our approach, we profiled the computational times for the pulley scene as a function of, n, the number of pulleys, with the C++ implementation. In this scene, each pulley is a separate rigid body expressed in maximal coordinates (6 DoF) with Lagrange multiplier constraints for the revolute joint (5 constraints). The computation time of such a scene would quickly be dominated by rigid body dynamics, since the total number of columns required for these rigid bodies in the system matrix is $(6 + 5) \times n$. Therefore, we modified the scene slightly, by removing the rigid bodies at the pulleys, and then passing the strand through a series of non-rotating frictionless pulleys that are attached alternatingly to the lower and upper beams.

With this modification to the pulley scene, we are able to simulate up to 4096 pulleys. The time required to simulate one second of the scene is shown in Fig. 9. Up to 128 pulleys can be simulated in real time. As we increase the number of pulleys, the cost of solving the linear system overpowers that of computing and filling the system matrices and vectors, and begins to dominate the overall cost. Asymptotically, the cost of filling the linear system increases linearly, while the cost of solving the linear system increases quadratically. The slopes of the log-log graphs are 1.088 and 1.941 respectively in the range $n \in [256, 4096]$. For larger matrices, iterative solvers such as preconditioned MINRES may yield better results. Further optimizations are possible, for example, by taking advantage of the sparsity pattern, or by multi-threading the matrix filling code.

6 Conclusions and Future Work

We described a very general framework that combines Lagrangian and Eulerian approaches, designed to handle the large-scale simulation of highly constrained strands.

There are some limitations with our approach. Finite friction with reduced coordinate models is non-trivial, and a topic of future work. Using the tension based resampling of Servin et al. [2010] is a straightforward extension, and should improve the robustness of the non-constrained portions of the strand. Also, our current implementation only supports viscous damping based on nodal velocities. The twist constraint formulation of Servin et al. [2007] and the quasistatic material frame formulation of Bergou et al. [2008] are possible extensions.

Our framework scales well for highly constrained strands. We achieved this with two key contributions. First, we introduced reduced nodes that have precisely the right degrees of freedom to match the constraints. Second, we introduced an Eulerian type of node, the E-node, into the standard Lagrangian formulation, which enabled us to handle constraints independently of the initial discretization of the strand. These contributions allowed us to simulate strands with smooth curve routing, accurate transmission of force directions, while avoiding locking artifacts.

Acknowledgments We would like to thank Danny Kaufman and the anonymous reviewers for their help. This work was made possible by the following sources of funding: the Canada Research Chairs Program, NSERC, CIHR, Human Frontier Science Program, and Peter Wall Institute for Advanced Studies.

References

Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., and Grinspun, E. 2008. Discrete elastic rods. *ACM Trans. Graph. 27*, 3 (Aug), 63:1–63:12.

Bergou, M., Audoly, B., Vouga, E., Wardetzky, M., and Grinspun, E. 2010. Discrete viscous threads. *ACM Trans. Graph. 29*, 4 (Jul), 116:1–116:10.

Bertails, F., Audoly, B., Cani, M.-P., Querleux, B., Leroy, F., and Lévêque, J.-L. 2006. Super-helices for predicting the dynamics of natural hair. *ACM Trans. Graph. 25*, 3 (Jul), 1180–1187.

BERTAILS, F. 2009. Linear time super-helices. *Computer Graphics Forum 28*, 2, 417–426.

BOYD, S., AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press.

CHENTANEZ, N., ALTEROVITZ, R., RITCHIE, D., CHO, L., HAUSER, K. K., GOLDBERG, K., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2009. Interactive simulation of surgical needle insertion and steering. *ACM Trans. Graph. 28*, 3 (Jul), 88:1–88:10.

CHOE, B., CHOI, M. G., AND KO, H.-S. 2005. Simulating complex hair with robust collision handling. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 153–160.

CLINE, M. B., AND PAI, D. K. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *Proc. IEEE International Conference on Robotics and Automation*, vol. 3, 3744–3751.

COLEMAN, P., AND SINGH, K. 2006. Cords: Geometric curve primitives for modeling contact. *IEEE Computer Graphics and Applications 26*, 3, 72–79.

DAVIS, T. A. 2006. *Direct Methods for Sparse Linear Systems*. SIAM Book Series on the Fundamentals of Algorithms. SIAM.

DELP, S. L., ANDERSON, F. C., ARNOLD, A. S., LOAN, P., HABIB, A., JOHN, T., GUENDELMAN, E., AND THELEN, D. G., 2007. Opensim: Open-source software to create and analyze dynamic simulations of movement.

DONEA, J., HUERTA, A., PONTHOT, J.-P., AND RODRIGUEZ-FERRAN, A. 2004. Arbitrary lagrangian-eulerian methods. *Encyclopedia of Computational Mechanics*.

FEATHERSTONE, R. 1987. *Robot Dynamics Algorithms*, 1st ed. Springer.

GARCÍA-FERNÁNDEZ, I., PLA-CASTELLS, M., AND MARTÍNEZ-DURÁ, R. J. 2008. Elevation cable modeling for interactive simulation of cranes. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 173–181.

GRÉGOIRE, M., AND SCHÖMER, E. 2007. Interactive simulation of one-dimensional flexible parts. *Comput. Aided Des. 39* (Aug), 694–707.

HADAP, S. 2006. Oriented strands: dynamics of stiff multi-body system. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 91–100.

KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2008. Simulating knitted cloth at the yarn level. *ACM Trans. Graph. 27*, 3 (Aug), 65:1–65:9.

KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2010. Efficient yarn-based cloth with adaptive contact linearization. *ACM Trans. Graph. 29*, 4 (Jul), 105:1–105:10.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph. 27*, 5 (Dec), 164:1–164:11.

KISS, B., LEVINE, J., AND MULLHAUPT, P. 1999. Modelling, flatness and simulation of a class of cranes. *Periodica Polytechnica, Electrical Engineering*, 43, 215–225.

KRY, P. G., AND PAI, D. K. 2003. Continuous contact simulation for smooth surfaces. *ACM Trans. Graph. 22*, 1 (Jan), 106–129.

KUBIAK, B., PIETRONI, N., GANOVELLI, F., AND FRATARCANGELI, M. 2007. A robust method for real-time thread simulation. In *Proc. ACM symposium on virtual reality software and technology*, 85–88.

LEE, S.-H., AND TERZOPOULOS, D. 2008. Spline joints for multibody dynamics. *ACM Trans. Graph. 27*, 3 (Aug), 22:1–22:8.

LENOIR, J., GRISONI, L., MESEURE, P., RÉMION, Y., AND CHAILLOU, C. 2004. Smooth constraints for spline variational modeling. In *GRAPHITE 2004*, 58–64.

LENOIR, J., COTIN, S., DURIEZ, C., AND NEUMANN, P. 2006. Interactive physically-based simulation of catheter and guidewire. *Computer & Graphics 30*, 3 (Jun), 417–423.

MARTIN, S., KAUFMANN, P., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2010. Unified simulation of elastic rods, shells, and solids. *ACM Trans. Graph. 29*, 4 (Jul), 39:1–39:10.

MCADAMS, A., SELLE, A., WARD, K., SIFAKIS, E., AND TERAN, J. 2009. Detail preserving continuum simulation of straight hair. *ACM Trans. Graph. 28*, 3 (Jul), 62:1–62:6.

MURRAY, R. M., LI, Z., AND SASTRY, S. S. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

NOCENT, O., AND REMION, Y. 2001. Continuous deformation energy for dynamic material splines subject to finite displacements. In *Proc. Eurographic workshop on Computer animation and simulation*, Springer-Verlag, 88–97.

PAI, D. K. 2002. Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum 21*, 3, 347–352.

QIN, H., AND TERZOPOULOS, D. 1996. D-NURBS: A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics 2*, 1, 85–96.

REMION, Y., NOURRIT, J., AND GILLARD, D. 1999. Dynamic animation of spline like objects. In *Proc. WSCG Conference*, 426–432.

SELLE, A., LENTINE, M., AND FEDKIW, R. 2008. A mass spring model for hair simulation. *ACM Trans. Graph. 27*, 3 (Aug), 64:1–64:11.

SERVIN, M., AND LACOURSIERE, C. 2007. Massless cable for real-time simulation. *Computer Graphics Forum 26*, 2, 172–184.

SERVIN, M., LACOURSIERE, C., AND BODIN, K. 2010. Hybrid, multi-resolution wires with massless frictional contacts. *IEEE Transactions on Visualization and Computer Graphics*.

SPILLMANN, J., AND TESCHNER, M. 2008. An adaptive contact model for the robust simulation of knots. *Computer Graphics Forum 27*, 2, 497–506.

SPILLMANN, J., AND TESCHNER, M. 2009. Cosserat nets. *IEEE Trans. Vis. Comput. Graph. 15*, 2, 325–338.

SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculotendon simulation for hand animation. *ACM Trans. Graph. 27*, 3 (Aug), 83:1–83:8.

THEETTEN, A., GRISONI, L., ANDRIOT, C., AND BARSKY, B. 2008. Geometrically exact dynamic splines. *Computer-Aided Design 40*, 1, 35 – 48.