Linear Time Super-Helices

Florence Bertails

EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Linear Time Super-Helices

Florence Bertails

BiPop - INRIA/LJK, France

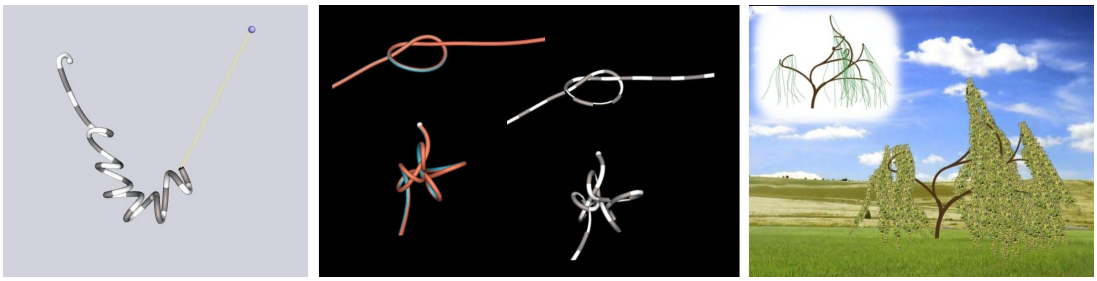

Figure 1: *Left: A Super-Helix of complex shape (composed of 30 helical elements) can be manipulated by the user in real-time thanks to our fast recursive algorithm. Middle: Complex configurations such as knots, involving a high resolution of the strand, can be achieved in real-time with our model. Right: Interactive animation of a weeping willow swept by wind, represented as a tree-like structure of Super-Helices (composed of 242 helical elements) with various stiffness values.*

Abstract

Thin elastic rods such as cables, phone coils, tree branches, or hair, are common objects in the real world but computing their dynamics accurately remains challenging. The recent Super-Helix model, based on the discrete equations of Kirchhoff for a piecewise helical rod, is one of the most promising models for simulating non-stretchable rods that can bend and twist. However, this model suffers from a quadratic complexity in the number of discrete elements, which, in the context of interactive applications, makes it limited to a few number of degrees of freedom - or equivalently to a low number of variations in curvature along the mean curve. This paper proposes a new, recursive scheme for the dynamics of a Super-Helix, inspired by the popular algorithm of Featherstone for serial multibody chains. Similarly to Featherstone's algorithm, we exploit the recursive kinematics of a Super-Helix to propagate elements inertias from the free end to the clamped end of the rod, while the dynamics is solved within a second pass traversing the rod in the reverse way. Besides the gain in linear complexity, which allows us to simulate a rod of complex shape much faster than the original approach, our algorithm makes it straightforward to simulate tree-like structures of Super-Helices, which turns out to be particularly useful for animating trees and plants realistically, under large displacements.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

1.1. Motivation

Realistic animations of hair, ropes, grass or foliage all require the accurate simulation of the same primitives: strands, *i.e.*, long and very thin deformable objects. Strands are ubiquitous in the real world, however their simulation remains challenging due to their specific nature: they can easily bend and twist, but they are unshearable and often resist stretching. Thin rods have been studied for more than one century in the field of elasticity theory - the reader can for example refer to [Ant95] for a modern derivation of these theories firstly developed by Euler, Cosserat and Kirchhoff. However, finding a proper discretization for the existing models is difficult, since the motion of thin elastic rods is governed by stiff equations.

1.2. Related work

Maximal *vs* generalized coordinates: One common approach for animating thin rods in computer graphics consists in using a *maximal coordinates* formulation of the system [Bar96] and enforcing the typical modes of deformation with external constraints. The advantage is to rely on simple chain models, such as the popular mass-spring system [RCT91], which are known to be fast and to have linear complexity in the number of simulated nodes. However, adding deformations constraints properly is difficult in the case of strand-like objects, for two main reasons. First, accounting for twisting requires a mathematical description of the orientation of the rod. Typically, mass-spring and 3D curve models such as splines [LMGC04] do not incorporate such an information. To account for twisting phenomena in hair simulation, Selle *et al.* attach additional springs to every three particle of the mass-spring chain [SLF08], while Theetten *et al.* add the rotation of the cross-section into the parameterization of the spline model and derive a corresponding twisting energy [TGAB08]. Secondly, enforcing the inextensibility constraint often leads to stiff systems and unstable simulations. Most previous approaches tolerate a bit of stretching in their models [PCP01, LMGC04, CCK05, ST07, TGAB08, SLF08], thus avoiding instabilities issues. In our work however, we are interested in perfectly inextensible models. Indeed, the inextensibility property is one of the causes for the stiff (or "nervous") response of moving strands, typically observable in long and straight hair, or perfect hair curls. Often, computational models incorporating stretching look too soft for capturing these effects accurately. Recently, Bergou *et al.* proposed to enforce the inextensibility constraint by relying on a fast projection method as a post-integration step, instead of computing a low-stretching energy and having to deal with stiff equations [BWR*08]. Though stable, this approach causes some artificial energy dissipation.

An alternate strategy to maximal coordinates formulation is to incorporate all the deformations constraints into the parameterization of the model, leading to a so-called *reduced coordinates* formulation. Serial chains of rigid segments are the most-well known reduced coordinate model for strands, and were successfully used for animating hair primitives [HMT01, CJY02, Had06]. However, each segment composing the strand being rigid, a high number of segments is required for animating strands with a non-straight shape at rest, leading to time-consuming simulations.

Quadratic complexity of the Super-Helix model: Recently, the Super-Helix model was proposed [BAC*06] for simulating the motion of individual hair strands in a realistic way, following Pai's work on the simulation of static Cosserat rods [Pai02]. Relying on a high-order discretization of the Kirchhoff equations for inextensible and unshearable elastic rods, the Super-Helix model captures the typical modes of deformation of real strands while being parameterized by only a few degrees of freedom: instead of using rigid segments as primitives, each element is represented as a deformable helix. However, the method suffers from a quadratic complexity in the number of elements composing the rod. This quadratic complexity is not relevant when animating free natural hair, where a low number of elements is sufficient to capture the deformations of hair realistically. However, as pointed out by Spillman *et al.*, it turns out to be an actual limitation when simulating high local deformations such as knots configurations, which typically occurs when the rod is subject to multiple contacts and self-collision [ST07]. In this paper, we propose a new, recursive algorithm for solving the dynamics of a Super-Helix in linear time. Our method is inspired by the popular algorithm by Featherstone [Fea83] for recursively solving the motion of serial multibody chains.

The Featherstone algorithm: Similarly to the Super-Helix model, computing the forward dynamics of a serial multibody chain amounts to solving a dense system, which implies a N^2 complexity in the case when the inertia matrix is constructed explicitly [FO00]. In the following, we shall qualify of *composite* [Fea87] such a method. Note however that the original, composite Super-Helix model did not benefit from any optimization in the computation of the mass matrix, unlike the so-called Composite-Rigid-Body algorithm [FO00]. The Articulated-Body approach [Fea83] was one of the first algorithms for computing the forward dynamics of articulated-bodies in linear time. This algorithm is explained in details in [Mir96]. The key of the approach relies on two features: first, the recursive nature of the kinematics and second, the observation that accelerations of the bodies are linear functions of the applied force. Therefore, cumulate inertias and force for a subchain can be precomputed within a first pass by propagating internal joint forces from the free end to the fixed end of the chain. Finally, the forward dynamics is recursively solved within a second pass traversing the structure in the reverse way. This algorithm was successfully adapted for solving the dynamics of articulated deformable bodies (see for instance [MA07]). To the best of our knowledge, it was never derived for a smooth, deformable model with no joint, such as the Super-Helix model.

1.3. Contributions

Our main contribution is to propose a new recursive algorithm for solving the dynamics of a Super-Helix in linear time. The key of our approach is to show that a dynamic equation can be locally formulated for each element, depending linearly on the acceleration of the first end of the element only. Similarly to the Featherstone algorithm, a first pass recursively computes the cumulate inertias and force from tip to root, and the forward dynamics is solved within a second pass traversing the strand from root to tip. A second contribution of this paper is the extension of this recursive algorithm for animating tree-like structures of Super-Helices. Interestingly, this method allows for the nonlinear simulation of stiff as well as soft foliage under large displacements.

1.4. Organization

The paper is organized as follows. Section 2 recalls the important notations for the Super-Helix model and introduces useful notations for an individual element. In section 3, we reformulate the kinematics and dynamics of a Super-Helix as recursive relationships and explain in details our new recursive algorithm in section 4. We derive the tree-like structure extension in section 5 before presenting results in section 6 and concluding.

2. Notations

For the sake of simplicity, we shall use similar notations as in [BAC*06] whenever appropriate. We shall introduce further notations only to deal with the kinematics of individual helical elements. In all our mathematical derivations, scalars, vectors and matrices will be denoted by using respectively plain (s), bold ($\mathbf{v}$) and blackboard bold ($\mathbb{M}$) fonts.

2.1. Notations for a Super-Helix

Let s be the curvilinear abscissa of the centerline of the Super-Helix, $0 \leq s \leq L$ where L is the total length of the rod. A Super-Helix is composed of N helical elements E_Q indexed by Q ($1 \leq Q \leq N$), each element being parameterized by a constant twist $\kappa_0 = \tau$ and two constant curvatures κ_1 and κ_2. The full Super-Helix is thus parameterized by the $3N$ generalized coordinates $\kappa_{i,Q}(t)$, where $i = 0,1,2$ specifies the mode of deformation for each element. The vector of size $3N$ collecting all the generalized coordinates is denoted $\boldsymbol{\kappa}(t)$, while the three functions $\kappa_i(s,t), i = 0,1,2$ stand for the piecewise constant twist and curvature functions along the rod:

$$\kappa_i(s,t) = \sum_{1 \leq Q \leq N} \chi_Q(s)\, \kappa_{i,Q}(t) \quad \text{for } i = 0,1,2, \qquad (1)$$

where $\chi_Q(s)$ equals 1 for $s \in E_Q$ and 0 elsewhere.

The curve $\mathbf{r}^{\mathrm{SH}}(s,t)$ gives the 3D position of the centerline at s. Each cross-section of the rod is associated with a material frame $\{\mathbf{n}_i^{\mathrm{SH}}(s,t)\}_{i=0,1,2}$ composed of a tangent vector $\mathbf{n}_0^{\mathrm{SH}}(s,t)$ and of two normal vectors $\mathbf{n}_1^{\mathrm{SH}}(s,t)$ and $\mathbf{n}_2^{\mathrm{SH}}(s,t)$ aligned with the principal axes of the cross-section. Note that unlike the twist and curvatures functions $\kappa_i(s,t)$, the centerline $\mathbf{r}^{\mathrm{SH}}(s,t)$ and the material frame $\{\mathbf{n}_i^{\mathrm{SH}}\}(s,t)_{i=0,1,2}$ are required to be respectively C^1 and C^0-smooth along the rod (*i.e.*, with respect to s). These smoothness conditions are essential for reconstructing the full geometry of a Super-Helix, given the vector of generalized coordinates $\boldsymbol{\kappa}$.

2.2. Notations for an element

The element of index Q is denoted $E_Q = \{s, s_{Q-1} \leq s \leq s_Q\}$. Note that with this definition, $s_0 = 0$ and $s_N = L$. Let ℓ_Q be the length of element E_Q, $\ell_Q = s_Q - s_{Q-1}$.

Let $f^{\mathrm{SH}}(s,t)$ be a function (scalar or vector) of the Super-Helix. One can decompose this function on each element as

$$f^{\mathrm{SH}}(s,t) = \sum_{1 \leq Q \leq N} \chi_Q(s)\, f_Q(s - s_{Q-1}, t)$$

where $f_Q(u,t)$ is the local function defined on E_Q and parameterized by $u = s - s_{Q-1} \in [0, \ell_Q]$, such that $f^{SH}(s,t) = f_Q(u,t)$ for $s \in E_Q$. Equation (1) is an example of such a decomposition. For a continuous function, we also have the condition

$$\forall Q \geq 2 \qquad f_Q(0) = f_{Q-1}(\ell_{Q-1}),$$

which helps reconstruct the function $f^{\mathrm{SH}}(s,t)$ if one is aware of a boundary condition, for example at $s = 0$ or $s = L$.

To alleviate the notations in the remainder of the paper, we shall omit variables other than u (such as the time variable t) in the description of local functions f_Q. Figure 2 summarizes the main notations used for an element.

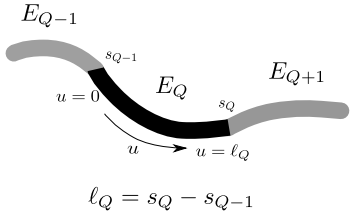

Figure 2: *Notations for an element E_Q.*

3. Towards Recursive Equations for a Super-Helix

The main equations for a Super-Helix are presented in [BAC*06] and reminded here to the reader. We also propose to formulate them as explicit recursive relationships in order to highlight the similarities between the Super-Helix model and articulated-bodies systems.

The Super-Helix is assumed to be clamped at the first end ($s = 0$) and free at the other end ($s = L$). In this section we also assume no external force is applied. The handling of external forces such as gravity will be addressed in section 4.5.

3.1. Recursive kinematics of a Super-Helix

The kinematics of the Super-Helix provides the space evolution of the centerline,

$$\frac{\partial \mathbf{r}^{\mathrm{SH}}}{\partial s}(s,t) = \mathbf{n}_0^{\mathrm{SH}}(s,t), \qquad (2a)$$

as well as the space evolution of the material frame,

$$\frac{\partial \mathbf{n}_i^{\mathrm{SH}}(s,t)}{\partial s} = \boldsymbol{\Omega}^{\mathrm{SH}}(s,t) \times \mathbf{n}_i^{\mathrm{SH}}(s,t) \quad \text{for } i = 0,1,2, \qquad (2b)$$

where $\boldsymbol{\Omega}^{\mathrm{SH}}(s,t) = \sum_{i=0,1,2} \kappa_i^{\mathrm{SH}}(t)\, \mathbf{n}_i^{\mathrm{SH}}(s,t)$ is the Darboux vector quantifying the spatial rate of rotation of the material frame along the rod. Note that equation (2a) expresses the

condition of inextensibility of the rod, since $\mathbf{n}_0$ is a unit vector. Note also that $\mathbf{\Omega}^{\mathrm{SH}}(s,t)$ is constant in space over each element. Thanks to this property, equations (2b) can be solved analytically through the reconstruction process [BAC*06]. An analytical expression for $\mathbf{r}^{\mathrm{SH}}(s,t)$ can then be derived using equation (2a), and subsequently all kinematic terms such as the time velocity $\dot{\mathbf{r}}^{\mathrm{SH}}(s,t)$ and acceleration $\ddot{\mathbf{r}}^{\mathrm{SH}}(s,t)$ of the centerline can be calculated analytically. The reader can refer to [Ber06] for a detailed and explicit formulation of these kinematic terms.

For the purpose of this paper, it will be sufficient to notice from [Ber06] that the accelerations $\ddot{\mathbf{r}}$ and $\ddot{\boldsymbol{\theta}}$ of the centerline and of the material frame respectively, can be formulated on each element E_Q as

$$\begin{cases} \ddot{\mathbf{r}}_Q(u) = \ddot{\mathbf{r}}_Q(0) + \mathbb{A}_Q^{\ddot{\mathbf{r}}}(u)\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{C}_Q^{\ddot{\mathbf{r}}}(u)\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}_Q^{\ddot{\mathbf{r}}}(u) \\ \ddot{\boldsymbol{\theta}}_Q(u) = \ddot{\boldsymbol{\theta}}_Q(0) + \mathbb{A}_Q^{\ddot{\boldsymbol{\theta}}}(u)\,\ddot{\boldsymbol{\kappa}}_Q + \mathbf{b}_Q^{\ddot{\boldsymbol{\theta}}}(u) \end{cases} \tag{3a}$$

where $\mathbb{A}_Q^{\ddot{\mathbf{r}}}(u)$, $\mathbb{C}_Q^{\ddot{\mathbf{r}}}(u)$, $\mathbf{b}_Q^{\ddot{\mathbf{r}}}(u)$, $\mathbb{A}_Q^{\ddot{\boldsymbol{\theta}}}(u)$ and $\mathbf{b}_Q^{\ddot{\boldsymbol{\theta}}}(u)$ only depend (non-linearly) on $\boldsymbol{\kappa}_Q$ and $\dot{\boldsymbol{\kappa}}_Q$, and not on any acceleration term. We also have the following smoothness conditions at the junction between elements

$$\begin{cases} \ddot{\mathbf{r}}_Q(0) = \ddot{\mathbf{r}}_{Q-1}(\ell_{Q-1}) \\ \ddot{\boldsymbol{\theta}}_Q(0) = \ddot{\boldsymbol{\theta}}_{Q-1}(\ell_{Q-1}) \end{cases} \quad \text{for } Q = 2,\cdots,N \tag{3b}$$

as well as the clamping conditions

$$\ddot{\mathbf{r}}_1(0) = \ddot{\mathbf{r}}_{cl} \qquad \ddot{\boldsymbol{\theta}}_1(0) = \ddot{\boldsymbol{\theta}}_{cl}. \tag{3c}$$

The set of equations (3) defines a *recursive* scheme for computing accelerations of a Super-Helix from its clamped end to its free end. Note that $\ddot{\mathbf{r}}_Q(u)$ and $\ddot{\boldsymbol{\theta}}_Q(u)$ *linearly* depend on every $\ddot{\boldsymbol{\kappa}}_{P,P\leq Q}$ where the dependence on $\ddot{\boldsymbol{\kappa}}_Q$ explicitly appears in equations (3a) and the dependence on $\ddot{\boldsymbol{\kappa}}_{P,P<Q}$ is implicitly contained in the terms $\ddot{\mathbf{r}}_Q(0)$ and $\ddot{\boldsymbol{\theta}}_Q(0)$ *via* the smoothness conditions (3b).

This recursive kinematics is very similar to the kinematics of an articulated-bodies system, as the same kind of linear dependency occurs between the global 3D accelerations of the system and the accelerations of its generalized coordinates. This property will be advantageously exploited for building our recursive Super-Helix algorithm (see section 4).

3.2. Recursive dynamics of a Super-Helix

In the absence of external forces and internal friction, the Lagrange equations of motion for a Super-Helix [BAC*06] read

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{\kappa}_{i,Q}}\right) - \frac{\partial T}{\partial \kappa_{i,Q}} + \frac{\partial U}{\partial \kappa_{i,Q}} = 0 \qquad \forall Q,i, \tag{4}$$

where

$$T(\boldsymbol{\kappa},\dot{\boldsymbol{\kappa}},t) = \frac{1}{2}\int_0^L \rho S \left(\dot{\mathbf{r}}^{\mathrm{SH}}(s,\boldsymbol{\kappa},t)\right)^2 \mathrm{d}s$$

is the kinetic energy of the rod, with ρS the mass per unit length, and

$$U(\boldsymbol{\kappa},t) = \frac{1}{2}\int_0^L \sum_{i=0}^{2} (EI)_i \left(\kappa_i^{\mathrm{SH}}(s,\boldsymbol{\kappa}) - \kappa_i^0(s)\right)^2 \mathrm{d}s$$

is the elastic potential energy of the rod. $(EI)_{i=0,1,2}$ are the twisting and bending moduli, assumed to be constant along the rod, and $\kappa_i^0(s)_{i=0,1,2}$ are the spontaneous twist and curvature functions, assumed to be piecewise constant in the same way as $\kappa_i^{\mathrm{SH}}(s,\boldsymbol{\kappa})$.

Let us detail the terms involved in equation (4). The gradient of the potential energy U can easily be derived as

$$\frac{\partial U}{\partial \kappa_{i,Q}} = \ell_Q\,(EI)_i\,(\kappa_{i,Q} - \kappa_{i,Q}^0),$$

where ℓ_Q is the length of element E_Q.

The exact derivation of the gradients of the kinetic energy can be worked out using the kinematics equations of the Super-Helix. We preferably rely on Maple for leading these calculations analytically, and skip details here for the sake of clarity. The first two terms of equation (4) finally read

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{\kappa}_{i,Q}}\right) - \frac{\partial T}{\partial \kappa_{i,Q}} = -\int_0^L \mathbf{T}(s,\boldsymbol{\kappa},t)\cdot\boldsymbol{\Phi}_{i,Q}(s,t)\,\mathrm{d}s \tag{5}$$

where $\mathbf{T}$ is the internal force of the rod such that

$$\rho S\,\ddot{\mathbf{r}}(s,\boldsymbol{\kappa},t) = \frac{\partial \mathbf{T}}{\partial s}(s,\boldsymbol{\kappa},t) \qquad \mathbf{T}(s=L) = 0 \tag{6}$$

and $\boldsymbol{\Phi}_{i,Q}$ a 3D function independent of $\boldsymbol{\kappa}$,

$$\boldsymbol{\Phi}_{i,Q}(s,t) = \left(\int_0^s \chi_Q(s')\,\mathbf{n}_i^{\mathrm{SH}}(s',t)\,\mathrm{d}s'\right) \times \mathbf{n}_0^{\mathrm{SH}}(s,t). \tag{7}$$

Note that $\boldsymbol{\Phi}_{i,Q}(s,t) = 0$ iff $s \in E_{P,P<Q}$, which is consistent with the fact that a perturbation $\delta\kappa_{i,Q}$ only affects elements $E_{R,R\geq Q}$.

Let $\boldsymbol{\varphi}_{i,Q}(u) = \int_0^u \mathbf{n}_{i,Q}(u')\,\mathrm{d}u'$. After replacing all the gradients and decomposing the right-hand side of (5) on each element E_Q, the Lagrange equations of motion (4) reread

$$(EI)_i\,(\kappa_{i,Q} - \kappa_{i,Q}^0) = \frac{1}{\ell_Q}\,(I_{i,Q} + J_{i,Q}) \qquad \forall Q,i \tag{8a}$$

where

$$\begin{cases} J_{i,Q} = \displaystyle\int_0^{\ell_Q} \mathbf{T}_Q(u)\cdot\left(\boldsymbol{\varphi}_{i,Q}(u)\times\mathbf{n}_{0,Q}(u)\right)\mathrm{d}u \\ I_{i,Q} = \displaystyle\sum_{R>Q}\int_0^{\ell_R} \mathbf{T}_R(u)\cdot\left(\boldsymbol{\varphi}_{i,Q}(\ell_Q)\times\mathbf{n}_{0,R}(u)\right)\mathrm{d}u, \end{cases} \tag{8b}$$

and we have the boundary condition

$$\mathbf{T}_N(\ell_N) = \mathbf{T}(L) = 0. \tag{8c}$$

Note that the left-hand side of equation (8a) only depends on the element E_Q while the right-hand side explicitly involves elements $E_{R,R\geq Q}$. Using this property as well as the free end condition (8c), we can expect to be able to reconstruct parts of the dynamics recursively, from the free end $Q = N$ to the clamped end $Q = 1$.

3.3. Composite *versus* recursive approach

Starting from the Lagrange equations of motion, the original Super-Helix method [BAC*06] builds a differential system for the unknowns $\boldsymbol{\kappa}$,

$$\mathbb{M}(t,\boldsymbol{\kappa})\,\ddot{\boldsymbol{\kappa}} + \mathbb{K}\,(\boldsymbol{\kappa}-\boldsymbol{\kappa}^0) = \mathbf{B}(t,\boldsymbol{\kappa},\dot{\boldsymbol{\kappa}}). \tag{9}$$

Unfortunately the inertia matrix $\mathbb{M}$ turns out to be dense, which can be explained by looking a bit more deeply at equations (8). Indeed, while the right-hand side of (8a) explicitly depends on elements $E_{R,R\geq Q}$, and thus only on accelerations $\ddot{\boldsymbol{\kappa}}_{R,R\geq Q}$, the internal force vector $\mathbf{T}_Q(u)$ implicitly depends on all other accelerations $\ddot{\boldsymbol{\kappa}}_{P,P<Q}$ through the linear momentum equation (6) and the kinematics equations (3). The dynamics equations (8) thus depend on *all* acceleration unknowns $\ddot{\boldsymbol{\kappa}}_{Q,1\leq Q\leq N}$, making the linear system dense and thus the solving algorithm of quadratic complexity.

Unlike this composite method, our approach avoids building the total inertia matrix $\mathbb{M}$, and instead exploits the recursive nature of both the kinematics (3) and the dynamics (8) of the Super-Helix model to achieve a linear solving algorithm. In the following, we show that we can actually reconstruct some parts of the right-hand side of (8a) from $Q=N$ to $Q=1$, the remaining terms being filled in within a second pass from $Q=1$ to $Q=N$.

4. A Recursive Algorithm for Solving the Dynamics

Let $G_{i,Q}=\frac{1}{\ell_Q}(I_{i,Q}+J_{i,Q})$ be the right-hand side of equation (8a). The key of our recursive algorithm is to show that $G_{i,Q}$ can be expressed locally on each element E_Q as a function of the 3 unknown accelerations $\ddot{\boldsymbol{\kappa}}_Q$, $\ddot{\mathbf{r}}_Q(0)$, and $\ddot{\boldsymbol{\theta}}_Q(0)$. In the following we call such a local expression for any 3D vector $\mathbf{f}$ a *decomposition* of $\mathbf{f}$ on the element E_Q. The decomposition of $\mathbf{f}$ on E_Q is formulated as the set of three 3×3 matrices $\mathbb{A}^{\mathbf{f}}_Q$, $\mathbb{R}^{\mathbf{f}}_Q$, $\mathbb{C}^{\mathbf{f}}_Q$ and one vector $\mathbf{b}^{\mathbf{f}}_Q$, such that[†]

$$\mathbf{f} = \mathbb{A}^{\mathbf{f}}_Q\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{R}^{\mathbf{f}}_Q\,\ddot{\mathbf{r}}_Q(0) + \mathbb{C}^{\mathbf{f}}_Q\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}^{\mathbf{f}}_Q.$$

Note that $\mathbb{A}^{\mathbf{f}}_Q$, $\mathbb{R}^{\mathbf{f}}_Q$, $\mathbb{C}^{\mathbf{f}}_Q$ and $\mathbf{b}^{\mathbf{f}}_Q$ may depend on u in case when $\mathbf{f}$ is a function of u (see for instance equations (3a)).

4.1. Induction hypothesis

We state that for all Q, $1\leq Q\leq N$, the right-hand side vector $\mathbf{G}_Q$ collecting the three scalars $(G_{i,Q})_{i=0,1,2}$ can be formulated as

$$\mathbf{G}_Q = \mathbb{A}^{\mathbf{G}}_Q\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{R}^{\mathbf{G}}_Q\,\ddot{\mathbf{r}}_Q(0) + \mathbb{C}^{\mathbf{G}}_Q\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}^{\mathbf{G}}_Q,$$

where the 3×3 matrices $\mathbb{A}^{\mathbf{G}}_Q$, $\mathbb{R}^{\mathbf{G}}_Q$ and $\mathbb{C}^{\mathbf{G}}_Q$ are called *cumulate inertias*, and the vector $\mathbf{b}^{\mathbf{G}}_Q$ *cumulate force* of the rod.

[†] In the following, we only use this notation for the decomposition of *local* functions $\mathbf{f}_Q$ on the element E_Q. For the sake of simplicity, we omit the index Q in the upperscript term $\mathbf{f}$ of the decomposition.

The above statement implies that the *local* dynamics equation on element E_Q read

$$\mathbb{K}_Q\,(\boldsymbol{\kappa}_Q-\boldsymbol{\kappa}^0_Q) = \mathbb{A}^{\mathbf{G}}_Q\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{R}^{\mathbf{G}}_Q\,\ddot{\mathbf{r}}_Q(0) + \mathbb{C}^{\mathbf{G}}_Q\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}^{\mathbf{G}}_Q \tag{10}$$

where $\mathbb{K}_Q$ is the stiffness matrix of element E_Q.

Note that our cumulate inertias and force extend the so-called *articulated-body inertia* and *bias force* introduced by Featherstone for serial multibody chains [Fea83], to the case of the Super-Helix model.

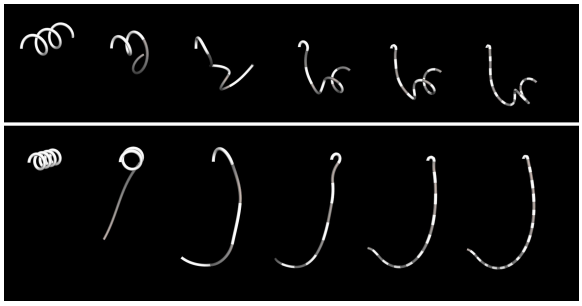

Figure 3: *Recursive Super-Helices in real-time, at various resolutions (1, 2, 5, 10, 20 and 30 elements from left to right). Top: snapshot of the rod falling under gravity. Bottom: snapshot of the rod swinging under oscillations of its root.*

4.2. First pass: accumulating inertias from tip to root

In appendix A we prove by mathematical induction that our local dynamics statement (10) holds for all elements, and we show that the cumulate inertias and force $\mathbb{A}^{\mathbf{G}}_Q$, $\mathbb{R}^{\mathbf{G}}_Q$, $\mathbb{C}^{\mathbf{G}}_Q$, $\mathbf{b}^{\mathbf{G}}_Q$ can be *recursively* computed from tip to root. The main step of the algorithm is actually to compute the decomposition of the internal forces $\mathbf{T}_{Q-1}(u)$ and $\mathbf{T}_Q(u)$ on element E_{Q-1} from the decomposition of $\mathbf{T}_Q(u)$ on element E_Q and from the cumulate inertias and force of element E_Q. Once we got this lower index decomposition, we can compute the decomposition of the right-hand side $\mathbf{G}_{Q-1}$ using the formulas (8b) for $J_{i,Q-1}$ and $I_{i,Q-1}$.

Below are the final recursive relationships for the decomposition of $\mathbf{T}_{Q-1}(u)$ on the element E_{Q-1}, useful for computing $J_{i,Q-1}$,

$$\begin{cases}
\mathbb{A}^{\mathbf{T}}_{Q-1}(u) = \mathbb{A}^{\mathbf{T}_{\mathbf{int}}}_{Q-1}(u) + \mathbb{T}_R\,\mathbb{A}^{\ddot{\mathbf{r}}}_{Q-1}(\ell_{Q-1}) + \mathbb{T}_C\,\mathbb{A}^{\ddot{\boldsymbol{\theta}}}_{Q-1}(\ell_{Q-1}) \\
\mathbb{R}^{\mathbf{T}}_{Q-1}(u) = \mathbb{R}^{\mathbf{T}_{\mathbf{int}}}_{Q-1}(u) + \mathbb{T}_R \\
\mathbb{C}^{\mathbf{T}}_{Q-1}(u) = \mathbb{C}^{\mathbf{T}_{\mathbf{int}}}_{Q-1}(u) + \mathbb{T}_R\,\mathbb{C}^{\ddot{\mathbf{r}}}_{Q-1}(\ell_{Q-1}) + \mathbb{T}_C \\
\mathbf{b}^{\mathbf{T}}_{Q-1}(u) = \mathbf{b}^{\mathbf{T}_{\mathbf{int}}}_{Q-1}(u) + \mathbb{T}_R\,\mathbf{b}^{\ddot{\mathbf{r}}}_{Q-1}(\ell_{Q-1}) + \mathbb{T}_C\,\mathbf{b}^{\ddot{\boldsymbol{\theta}}}_{Q-1}(\ell_{Q-1}) \\
\qquad\qquad + \mathbf{t}_b + \mathbb{A}^{\mathbf{T}}_Q(0)\left(\mathbb{A}^{\mathbf{G}}_Q\right)^{-1}\left(\mathbb{K}_Q\,(\boldsymbol{\kappa}_Q-\boldsymbol{\kappa}^0_Q)\right)
\end{cases}$$

where the (non-recursive) decomposition of $\mathbf{T}_{\mathbf{int}\,Q-1}(u)$ on E_{Q-1} is given by equations (15) in appendix A, and $\mathbb{T}_R$, $\mathbb{T}_C$ and $\mathbf{t}_b$ are given by

$$\begin{cases} \mathbb{T}_R = -\mathbb{A}_Q^{\mathbf{T}}(0)\left(\mathbb{A}_Q^{\mathbf{G}}\right)^{-1}\mathbb{R}_Q^{\mathbf{G}} + \mathbb{R}_Q^{\mathbf{T}}(0) \\ \mathbb{T}_C = -\mathbb{A}_Q^{\mathbf{T}}(0)\left(\mathbb{A}_Q^{\mathbf{G}}\right)^{-1}\mathbb{C}_Q^{\mathbf{G}} + \mathbb{C}_Q^{\mathbf{T}}(0) \\ \mathbf{t}_b = -\mathbb{A}_Q^{\mathbf{T}}(0)\left(\mathbb{A}_Q^{\mathbf{G}}\right)^{-1}\mathbf{b}_Q^{\mathbf{G}} + \mathbf{b}_Q^{\mathbf{T}}(0). \end{cases}$$

The formulas for the decomposition of $\mathbf{T}_Q(u)$ on E_{Q-1}, useful for computing $I_{i,Q-1}$, are similar to the ones above.

4.3. Second pass: solving the dynamics from root to tip

Once the first pass is processed, we know all cumulate inertias and force $\mathbb{A}_Q^{\mathbf{G}}$, $\mathbb{R}_Q^{\mathbf{G}}$, $\mathbb{C}_Q^{\mathbf{G}}$, $\mathbf{b}_Q^{\mathbf{G}}$ for $1 \le Q \le N$. In the dynamics equation (10), two parameters remain unknown: $\ddot{\mathbf{r}}_Q(0)$ and $\ddot{\boldsymbol{\theta}}_Q(0)$. These two parameters can however be reconstructed from root to tip according to the kinematics equations (3). We can thus recursively solve the system (10) for $\boldsymbol{\kappa}$ from root to tip. Here is our detailed algorithm :

- Initialization: set $\mathbf{r}_1(0,t+\delta t)$, $\dot{\mathbf{r}}_1(0,t+\delta t)$, $\ddot{\mathbf{r}}_1(0,t+\delta t)$, $\boldsymbol{\theta}_1(0,t+\delta t)$, $\dot{\boldsymbol{\theta}}_1(0,t+\delta t)$ and $\ddot{\boldsymbol{\theta}}_1(0,t+\delta t)$ to their clamped values at time $t+\delta t$.
- Loop, for $Q=1$ to $Q=N$:
  1. Build the 3×3 differential equation for $\boldsymbol{\kappa}_Q$
  $$\mathbb{M}_Q(t,\boldsymbol{\kappa})\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{K}_Q\,(\boldsymbol{\kappa}_Q - \boldsymbol{\kappa}_Q^0) = \mathbf{B}_Q(t,\boldsymbol{\kappa},\dot{\boldsymbol{\kappa}}),$$
  where $\mathbb{M}_Q = -\mathbb{A}_Q^{\mathbf{G}}$, $\mathbf{B}_Q = \mathbb{R}_Q^{\mathbf{G}}\ddot{\mathbf{r}}_Q(0) + \mathbb{C}_Q^{\mathbf{G}}\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}_Q^{\mathbf{G}}$.
  2. Solve the equation using a semi-implicit scheme. We obtain the new local twist and curvatures $\boldsymbol{\kappa}_Q(t+\delta t)$ as well as their derivatives $\dot{\boldsymbol{\kappa}}_Q(t+\delta t)$.
  3. Build the kinematics of element E_Q from $\boldsymbol{\kappa}_Q(t+\delta t)$, $\dot{\boldsymbol{\kappa}}_Q(t+\delta t)$, $\mathbf{r}_Q(0,t+\delta t)$, $\dot{\mathbf{r}}_Q(0,t+\delta t)$, $\boldsymbol{\theta}_Q(0,t+\delta t)$ and $\dot{\boldsymbol{\theta}}_Q(0,t+\delta t)$.
  4. Compute $\mathbf{r}_Q(\ell_Q,t+\delta t)$, $\dot{\mathbf{r}}_Q(\ell_Q,t+\delta t)$, $\ddot{\mathbf{r}}_Q(\ell_Q,t+\delta t)$, $\boldsymbol{\theta}_Q(\ell_Q,t+\delta t)$, $\dot{\boldsymbol{\theta}}_Q(\ell_Q,t+\delta t)$ and $\ddot{\boldsymbol{\theta}}_Q(\ell_Q,t+\delta t)$ using the kinematics of element E_Q.
  5. Deduce $\mathbf{r}_{Q+1}(0,t+\delta t)$, $\dot{\mathbf{r}}_{Q+1}(0,t+\delta t)$, $\ddot{\mathbf{r}}_{Q+1}(0,t+\delta t)$, $\boldsymbol{\theta}_{Q+1}(0,t+\delta t)$, $\dot{\boldsymbol{\theta}}_{Q+1}(0,t+\delta t)$ and $\ddot{\boldsymbol{\theta}}_{Q+1}(0,t+\delta t)$ from the previous step using the smoothness conditions between elements Q and $Q+1$.

4.4. Reduction of the system

As raised in [Ber06], the mass matrix $\mathbb{M}$ in the general Super-Helix equation (9) is nearly of rank $2N$ in a $3N$ system. This is due to the assumption of the infinitesimal cross-section area of the rod, which causes the system to be degenerate. This singularity is not significant in the original composite solving method presented in [BAC*06], since the final matrix to be inverted is $\mathbb{M}+dt^2\,\mathbb{K}$ which is non-singular provided the chosen time step is not too small. However, our recursive approach requires the inversion of the mass matrix $\mathbb{A}_Q$ in equation (16). To ensure that $\mathbb{A}_Q$ is non-singular, we consider that the temporal evolution of the twist $\kappa_{0,Q}$ is negligible, and simply freeze its value to the spontaneous twist $\kappa_{0,Q}^0$. This assumption is actually consistent with the recent analysis provided by [BWR*08] mentioning that twist waves propagate much faster than bending waves in a thin rod, and even instantly at the limit of a vanishing cross-section. Finally, we only keep the two actual degrees of freedom $\kappa_{1,Q}$ and $\kappa_{2,Q}$ as unknowns of our equations, and we solve 2×2 non-singular systems at each time step (using direct matrix inversion). Our results show that this simplification has no significant visual impact on the simulation of a Super-Helix, compared to the original algorithm.

4.5. External forces

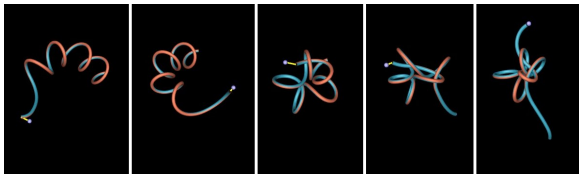

Figure 4: *The user is interactively manipulating the free end of a highly curly rod composed of 30 elements and subject to self-contacts.*

External forces $\mathbf{F}(s,t)$ such as gravity, air friction or contact forces can be introduced within our algorithm by simply adding their contribution to the expression of the internal force $\mathbf{T}(s)$ as

$$\mathbf{T}(s,t) = \int_s^L \left(\mathbf{F}(s',t) - \rho\, S\,\ddot{\mathbf{r}}(s',\boldsymbol{\kappa},t)\right) \mathrm{d}s'.$$

For example, in the case of gravity, $\mathbf{F} = \rho\, S\,\mathbf{g}$ and

$$\mathbf{T}(s,t) = \rho\, S\left(\mathbf{g}(L-s) - \int_s^L \ddot{\mathbf{r}}(s',\boldsymbol{\kappa},t)\,\mathrm{d}s'\right). \qquad (11)$$

Note that adding a punctual force $\mathbf{F}_{\mathrm{L}}$ at the free end of the rod (see figure 4) is simply achieved by modifying the boundary condition (8c) in the first pass as

$$\mathbf{T}_N(\ell_N) = \mathbf{F}_{\mathrm{L}}.$$

In practice we also add some internal friction to the model similarly as in [BAC*06] so that the rod is slightly damped even in the absence of air friction. This is simply done by adding the term $-\nu\,\mathbb{K}\,\dot{\boldsymbol{\kappa}}_Q$ into the final linear system, and by correcting the induction hypothesis (10) accordingly.

4.6. Featherstone's gravity

Within the recursive Articulated-Body algorithm [Fea87], Featherstone demonstrated that gravity can be efficiently accounted for by modifying the acceleration of the root element during the second pass, instead of projecting the contribution of gravity onto the frame of each element during the first pass.

This method can directly be transposed onto our recursive Super-Helix algorithm. Indeed, the contribution of gravity on element E_Q, already given in equation (11), can be reformulated as

$$\mathbf{T}_Q(u) = \rho S \int_u^{\ell_Q} \left(\mathbf{g} - \ddot{\mathbf{r}}_Q(u')\right) \mathrm{d}u'.$$

We can then compute the modified decomposition of $\mathbf{T}_Q$ on $\ddot{\boldsymbol{\kappa}}_Q$, $\ddot{\mathbf{r}}'_Q(0)$, $\ddot{\boldsymbol{\theta}}_Q(0)$ where $\ddot{\mathbf{r}}'_Q(0)$ is the modified acceleration $\ddot{\mathbf{r}}_Q(0) - \mathbf{g}$. The first pass of the recursive algorithm thus remains unchanged, except that we do not compute the gravity contribution for each element anymore. During the second pass, the gravity component has to be incorporated into each linear acceleration $\ddot{\mathbf{r}}_Q(0)$. Since we recursively compute the next acceleration $\ddot{\mathbf{r}}_{Q+1}(u)$ using the smoothness condition (3b) for $\dddot{\mathbf{r}}_Q(u)$ at $u = \ell_Q$, we only need to add the contribution of gravity to the root element E_1. That is, the only change to make in the second pass for applying gravity is to modify the initialization of $\ddot{\mathbf{r}}_1(0)$ as

$$\ddot{\mathbf{r}}'_1(0) = \ddot{\mathbf{r}}_1(0) - \mathbf{g}.$$

In practice, we noticed that when using the Featherstone's gravity "trick", simulations are slightly faster. For example, the animation of a Super-Helix composed of 50 elements is running at 15 FPS when using the Featherstone's gravity method, compared with 14.5 FPS when using the regular way of adding gravity.

5. Extension to Tree-like Super-Helices

While animating a tree-like structure of Super-Helices would be tricky using the composite method, this becomes straightforward with our recursive algorithm. Indeed, the only change to make for building the cumulate inertias and force is to set a proper boundary condition for $\mathbf{T}_Q(\ell_Q)$ when computing $J_{i,Q}$ from the induction hypothesis at upper index. Let $\mathscr{T}^{SH}$ be a tree of Super-Helices. We call $\mathscr{D}_Q$ the set of indices of the elements that are descendants of the element E_Q in the tree (greater index). Note that in the case when $\mathscr{T}^{SH}$ is a plain Super-Helix with no branches (*i.e.*, a degenerate tree), $\mathscr{D}_Q$ is reduced to the singleton $\{Q+1\}$ for all Q, $1 \leq Q \leq N$ and $\mathbf{T}_Q(\ell_Q) = \mathbf{T}_{Q+1}(0)$ as set in equation (14) of appendix A and in the computation of $J_{i,Q}$.

For a general tree-like structure, the new boundary condition for $\mathbf{T}_Q$ reads

$$\mathbf{T}_Q(\ell_Q) = \sum_{R \in \mathscr{D}_Q} \mathbf{T}_R(0).$$

The first pass of the algorithm starts from all the leaves of the tree and recursively propagates computations of cumulate inertias and force from a set of descendants to their (common) ascendant until it reaches the root.

As for the second pass, we simply need to propagate the kinematics values computed at the last end of element E_Q to the first end of all descendants of E_Q. Basically, only step 5 of the algorithm in 4.3 needs to be changed by replacing $Q+1$ whenever it appears by $R \in \mathscr{D}_Q$. Note that the second pass starts from the root of the tree, recursively propagates towards the various branches, and finally stops at the leaves.

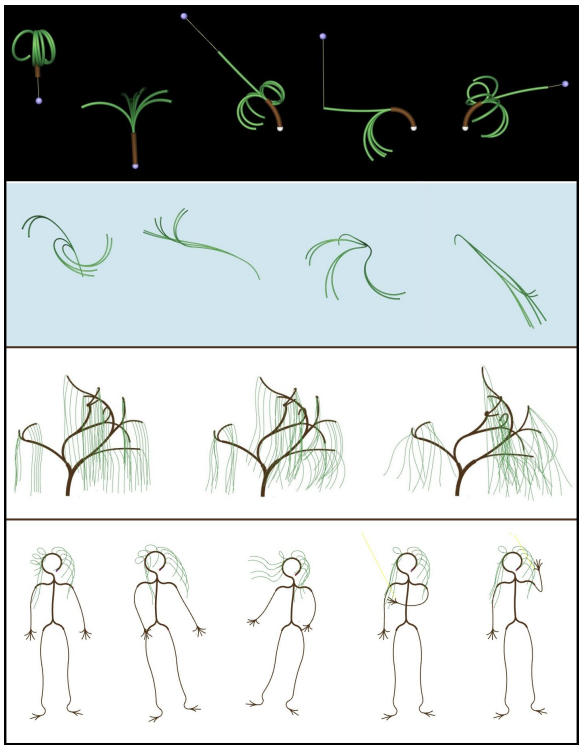

Figure 5: *Interactive animation of various trees made of Super-Helices. From top to bottom: palm tree manipulated by the user (8 elements), algae deforming in a water flow (10 elements), weeping willow swept by wind (242 elements), Super-Helix man manipulated by the user (81 elements).*

6. Results and Discussion

6.1. Recursive Super-Helix

Figures 1, left and middle, and 3, 4, as well as the first part of the accompanying video, present a number of examples of rods (with various spontaneous curvatures) simulated with our recursive method, as well as visual comparisons with the original composite approach. The first part of the video especially demonstrates both the efficiency and the stability of the algorithm for a high number of elements.

6.2. Performance

As shown in table 1, our recursive algorithm scales favorably compared to the composite approach. Note that the performance of the composite method drastically decreases when $N > 10$ elements, whereas our method keeps on being fast

for a high number of elements. Our results were processed on a single threaded application running on an Intel Core 2 Duo CPU at 2.1 GHz.

Model (N elements)	FPS recursive	FPS composite
Straight rod (1)	170	170
Straight rod (2)	170	170
Straight rod (10)	120	40
Straight rod (20)	54	4
Straight rod (30)	37	1.4
Palm tree (8)	170	-
Algae (10)	108	-
Complex 3D tree (50)	25	-
Super-Helix man (81)	17	-
Weeping willow (242)	6	-

Table 1: *Comparisons of performance (in frames per second) between the recursive and the composite methods, for the examples presented in the accompanying video.*

6.3. Stability

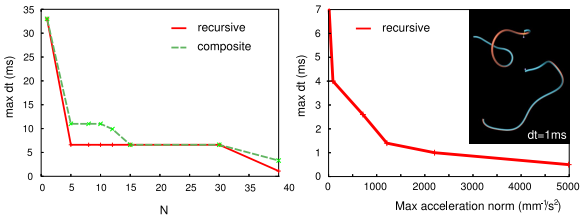

Figure 6: *Stability of our algorithm: Measure of the maximal time step allowing for convergence, depending on (left) the number N of elements of the rod (measurements performed on the motions of figure 3), and (right) the maximum acceleration of the rod during motion (computed as max$\|\ddot{\boldsymbol{\kappa}}\|$ over the time steps). The right image illustrates the ability of our approach to simulate "wild" motions in a stable way when decreasing the time step - such motions being difficult to achieve with the composite method.*

Figure 6 reports a few measurements that we have performed for analyzing the influence of both the number N of elements of the rod and the acceleration of the rod on stability. The left figure shows that the recursive algorithm generally requires a smaller time step than the composite approach. This is due to the explicit computation of $\ddot{\boldsymbol{\kappa}}_Q$ through equation (16) in the first pass of our algorithm, which makes it slightly less stable than the composite method for an equivalent time step. In the case of articulated rigid bodies, the explicit solving of Featherstone's algorithm was pointed out by Hadap [Had03] who recently proposed an interesting framework for enforcing an implicit solving [Had06]. Note however that in our case, the difference in stability between the two methods keeps on being small. This limitation is also mitigated by the fact that for "wild" motions involving high and fast deformations of the rod, our recursive approach is actually more stable than the composite method: as illustrated in the right figure, our method can still handle strong generalized accelerations of the rod at reasonable frame rates (a few seconds per frame) whereas such scenarios (that require small time steps) are hardly achievable using the original composite approach, as decreasing the time step would lead to an ill-defined system (see section 4.4).

6.4. Animation of tree-like structures

Figures 1, right, and 5, 7, as well as the second part of the accompanying video show some animations of tree-like Super-Helices. They illustrate the benefits of our method compared to classical approaches for animating trees:

- Compared to approaches based on modal analysis [Sta97], our model captures the *nonlinear* deformations of a tree-like structure, while being as low time consuming as other approaches. It is thus suited for handling large displacements of tree-like structures, which in practice turns out to be very useful for animating soft foliages such as weeping willows, algae, or any tree subject to wind gust. In our examples, we simulate rigid as well as soft tree-like structures, subject to high deformations. Such systems could hardly be simulated realistically using traditional modal analysis.
- Unlike some interactive methods [SO99], our algorithm is based on an actual *bilateral* traversal of the structure for updating accelerations of our tree-like structure, from the root to the leaves and from the leaves to the root. This means that a local perturbation of the tree affects both the descendants of the perturbed element *and* its ascendants. Our model is thus suited for handling external forces (such as contacts or wind force) properly.
- As shown by our results, we benefit from the compact representation of the Super-Helix model for providing a new and versatile dynamic tree model whose parameters (mass, stiffness, ...) are very intuitive and easy to tune. This allows for the quick generation of various types of animations of a given tree (see for example figure 7).

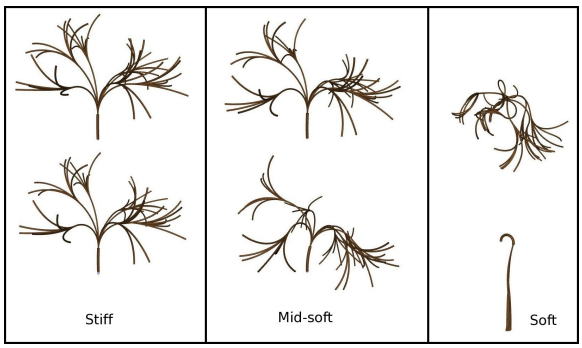

Figure 7: *Versatility of our tree model: a complex tree with various stiffness values falling under gravity (50 elements).*

Performance obtained for the examples of figures 1, right, and 5, 7 are shown in table 1. Compared to the recent method by [BWR*08] that reports a time cost of 22 ms for a 0.1 ms time step on a complex tree example, our method runs more than 50 times faster when animating a tree with a similar complexity (such as the weeping willow). This is first due to the good stability of our algorithm, which allows for the use of large time steps (11 ms) even when simulating complex objects. Furthermore, using a high-order representation of branches enables us to get accurate results by simulating only a few hundred elements instead of thousands of nodes.

7. Conclusion

We have presented a new, recursive algorithm for solving the dynamics of the Super-Helix model for strand simulation. Our algorithm particularly improves the efficiency of the original method when simulating long and complex rods made of more than 5 elements, and also proves well-suited for efficiently animating tree-like structures under large displacements. In the future, we wish to exploit our recursive algorithm for designing adaptive dynamic schemes of rods and trees, in the same spirit as [RGL05] and [BKCN03, WL03], while using a high-order strand primitive. We would also like to investigate the problem of contacts, following the accurate constraints-based methods developed for serial multibody chains [Kok04, Had06].

Acknowledgments The author would like to thank Basile Audoly for sharing with her the code for the kinematics of a Super-Helix. She is also grateful to the anonymous reviewers for their useful comments.

References

[Ant95] ANTMAN S.: *Nonlinear Problems of Elasticity*. Springer Verlag, 1995.

[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. In *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH'06 conference)* (2006), pp. 1180–1187.

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'96 conference)* (1996), pp. 137–146.

[Ber06] BERTAILS F.: *Simulation of Virtual Hair*. PhD thesis, Institut National Polytechnique de Grenoble, June 2006.

[BKCN03] BERTAILS F., KIM T.-Y., CANI M.-P., NEUMANN U.: Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. In *ACM SIGGRAPH - EG Symposium on Computer Animation (SCA'03)* (July 2003), pp. 207–213.

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete elastic rods. *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH'08 conference) 27*, 3 (2008), 1–12.

[CCK05] CHOE B., CHOI M., KO H.-S.: Simulating complex hair with robust collision handling. In *ACM SIGGRAPH - EG Symposium on Computer Animation (SCA'05)* (August 2005), pp. 153–160.

[CJY02] CHANG J., JIN J., YU Y.: A practical model for hair mutual interactions. In *ACM SIGGRAPH - EG Symposium on Computer Animation (SCA'02)* (July 2002), pp. 73–80.

[Fea83] FEATHERSTONE R.: The calculation of robot dynamics using articulated-body inertias. *International Journal of Robotics Research 2*, 1 (1983), 13–30.

[Fea87] FEATHERSTONE R.: *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[FO00] FEATHERSTONE R., ORIN D.: Robot dynamics: equations and algorithms. *IEEE International Conference on Robotics and Automation* (2000), 826–834.

[Had03] HADAP S.: *Hair Simulation*. PhD thesis, Université de Genève, January 2003.

[Had06] HADAP S.: Oriented strands - dynamics of stiff multibody system. In *ACM SIGGRAPH - EG Symposium on Computer Animation (SCA'06)* (2006), pp. 91–100.

[HMT01] HADAP S., MAGNENAT-THALMANN N.: Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3 (2001), 329–338. Proceedings of Eurographics'01.

[Kok04] KOKKEVIS E.: Practical physics for articulated characters. In *Proceedings of Game Developers Conference* (2004).

[LMGC04] LENOIR J., MESEURE P., GRISONI L., CHAILLOU C.: A suture model for surgical simulation. *2nd International Symposium on Medical Simulation (ISMS'04)* (June 2004), 105–113.

[MA07] MUKHERJEE R., ANDERSON K.: A logarithmic complexity divide-and-conquer algorithm for multi-flexible articulated body dynamics. *Journal of Computational and Nonlinear Dynamics 2*, 1 (2007), 10–21.

[Mir96] MIRTICH B.: *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, Berkeley, 1996.

[Pai02] PAI D.: Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum 21*, 3 (2002), 347–352. Proceedings of Eurographics'02.

[PCP01] PLANTE E., CANI M.-P., POULIN P.: A layered wisp model for simulating interactions inside long hair. In *EG workshop on Computer Animation and Simulation (EG CAS'01)* (Sept. 2001), Computer Science, Springer, pp. 139–148.

[RCT91] ROSENBLUM R., CARLSON W., TRIPP E.: Simulating the structure and dynamics of human hair: Modeling, rendering, and animation. *The Journal of Visualization and Computer Animation 2*, 4 (1991), 141–148.

[RGL05] REDON S., GALOPPO N., LIN M.: Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH'05 conference) 24*, 3 (2005), 936–945.

[SLF08] SELLE A., LENTINE M., FEDKIW R.: A mass spring model for hair simulation. *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH'08 conference) 27*, 3 (2008), 1–11.

[SO99] SAKAGUCHI T., OHYA J.: Modeling and animation of botanical trees for interactive virtual environments. In *VRST '99: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 1999), ACM, pp. 139–146.

[ST07] SPILLMANN J., TESCHNER M.: Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *ACM SIGGRAPH - EG Symposium on Computer Animation (SCA'07)* (2007), pp. 63–72.

[Sta97] STAM J.: Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum 16*, 3 (1997), 159–164.

[TGAB08] THEETTEN A., GRISONI L., ANDRIOT C., BARSKY B.: Geometrically exact splines. *Journal of Computer Aided Design 40*, 1 (2008), 35–48.

[WL03] WARD K., LIN M.: Adaptive grouping and subdivision for simulating hair dynamics. In *Proceedings of Pacific Graphics'03* (Sept. 2003), pp. 234–243.

Appendix A: Proof of the First Pass

We demonstrate here that the induction hypothesis (10) holds for every Q, $1 \leq Q \leq N$, and we show how to build the cumulate inertias and forces from the free end to the clamped end of the Super-Helix. For that purpose, we shall prove the following relationships first

$$\begin{aligned}
\mathbf{T}_{\mathbf{int}Q}(u) &= \mathbb{A}_Q^{\mathbf{T}_{\mathbf{int}}}(u)\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{R}_Q^{\mathbf{T}_{\mathbf{int}}}(u)\,\ddot{\mathbf{r}}_Q(0) + \mathbb{C}_Q^{\mathbf{T}_{\mathbf{int}}}(u)\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}_Q^{\mathbf{T}_{\mathbf{int}}}(u)\\
\mathbf{T}_Q(u) &= \mathbb{A}_Q^{\mathbf{T}}(u)\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{R}_Q^{\mathbf{T}}(u)\,\ddot{\mathbf{r}}_Q(0) + \mathbb{C}_Q^{\mathbf{T}}(u)\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}_Q^{\mathbf{T}}(u)\\
J_{i,Q} &= \mathbf{a}_{i,Q}^J \cdot \ddot{\boldsymbol{\kappa}}_Q + \mathbf{r}_{i,Q}^J \cdot \ddot{\mathbf{r}}_Q(0) + \mathbf{c}_{i,Q}^J \cdot \ddot{\boldsymbol{\theta}}_Q(0) + b_{i,Q}^J\\
\mathbf{L}_Q &= \mathbb{A}_Q^{\mathbf{L}}\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{R}_Q^{\mathbf{L}}\,\ddot{\mathbf{r}}_Q(0) + \mathbb{C}_Q^{\mathbf{L}}\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}_Q^{\mathbf{L}}\\
I_{i,Q} &= \mathbf{a}_{i,Q}^I \cdot \ddot{\boldsymbol{\kappa}}_Q + \mathbf{r}_{i,Q}^I \cdot \ddot{\mathbf{r}}_Q(0) + \mathbf{c}_{i,Q}^I \cdot \ddot{\boldsymbol{\theta}}_Q(0) + b_{i,Q}^I,
\end{aligned} \tag{12}$$

where we have defined the following intermediary quantity

$$\mathbf{T}_{\mathbf{int}Q}(u) = -\int_u^{\ell_Q} \rho\, S\, \ddot{\mathbf{r}}_Q(u')\mathrm{d}u'. \tag{13}$$

By setting $\mathbf{T}_{N+1}(0) = \mathbf{T}_N(\ell_N) = 0$, $\mathbf{T}_Q(u)$ can be reconstructed from tip to root using the following recursive relationship

$$\mathbf{T}_Q(u) = \mathbf{T}_{Q+1}(0) + \mathbf{T}_{\mathbf{int}Q}(u) \quad \text{for } 1 \leq Q \leq N. \tag{14}$$

We have also introduced the following quantity:

$$\mathbf{L}_Q = \sum_{R>Q} \int_0^{\ell_R} \mathbf{T}_R(u) \times \mathbf{n}_{0,R}(u)\,\mathrm{d}u.$$

First note that proving the recursive relationship for $\mathbf{T}_{\mathbf{int}Q}(u)$ is straightforward, as $\mathbf{T}_{\mathbf{int}Q}(u)$ only depends on the element E_Q. Using equations (13) and (3a) we have

$$\forall Q,\, 1 \leq Q \leq N \qquad \begin{cases}
\mathbb{A}_Q^{\mathbf{T}_{\mathbf{int}}}(u) = -\rho\, S \displaystyle\int_u^{\ell_Q} \mathbb{A}_Q^{\ddot{\mathbf{r}}}(u')\,\mathrm{d}u'\\
\mathbb{R}_Q^{\mathbf{T}_{\mathbf{int}}}(u) = \rho\, S\,(u - \ell_Q)\\
\mathbb{C}_Q^{\mathbf{T}_{\mathbf{int}}}(u) = -\rho\, S \displaystyle\int_u^{\ell_Q} \mathbb{C}_Q^{\ddot{\mathbf{r}}}(u')\,\mathrm{d}u'\\
\mathbf{b}_Q^{\mathbf{T}_{\mathbf{int}}}(u) = -\rho\, S \displaystyle\int_u^{\ell_Q} \mathbf{b}_Q^{\ddot{\mathbf{r}}}(u')\,\mathrm{d}u'.
\end{cases} \tag{15}$$

Index N

In that case, $\mathbf{T}_N(u) = \mathbf{T}_{\mathbf{int}N}(u)$, $\mathbf{L}_N = \mathbf{0}$, and $I_{i,N} = 0$. We have, for $i = 0, 1, 2$,

$$G_{i,N} = \frac{1}{\ell_N} J_{i,N} = \int_0^{\ell_N} \mathbf{T}_{\mathbf{int}N}(u) \cdot (\boldsymbol{\varphi}_{i,N}(u) \times \mathbf{n}_{0,N}(u))\,\mathrm{d}u.$$

The cumulate inertias and force $\mathbb{A}_Q^{\mathbf{G}}$, $\mathbb{R}_Q^{\mathbf{G}}$, $\mathbb{C}_Q^{\mathbf{G}}$, $\mathbf{b}_Q^{\mathbf{G}}$ are thus deduced from the components of $\mathbf{T}_{\mathbf{int}N}$, given in equations (15), by the dot product with a known function (independent of $\ddot{\boldsymbol{\kappa}}_Q$, $\ddot{\mathbf{r}}_Q(0)$ and $\ddot{\boldsymbol{\theta}}_Q(0)$), and by integration on the element E_N.

Index Q-1

We now assume the induction hypothesis (10) and (12) are true at index Q. We aim at showing that they still hold at index $Q-1$, and give insights for building the cumulate inertias and force at index $Q-1$.

Let us first have a look at $J_{i,Q-1}$.

$$J_{i,Q-1} = \int_0^{\ell_{Q-1}} \mathbf{T}_{Q-1}(u) \cdot \left(\boldsymbol{\varphi}_{i,Q-1}(u) \times \mathbf{n}_{0,Q-1}(u)\right) \mathrm{d}u$$

where $\mathbf{T}_{Q-1}(u) = \mathbf{T}_Q(0) + \mathbf{T}_{\mathbf{int}Q-1}(u)$ through the smoothness condition (14). Note that the only one part of $J_{i,Q-1}$ that is not explicitly depending on $\ddot{\boldsymbol{\kappa}}_{Q-1}$, $\ddot{\mathbf{r}}_{Q-1}(0)$, $\ddot{\boldsymbol{\theta}}_{Q-1}(0)$ is coming from the integration of the dot product by $\mathbf{T}_Q(0)$, since $\mathbf{T}_{\mathbf{int}Q-1}(u)$ can be decomposed on the element E_{Q-1} through equation (15). We thus need to decompose $\mathbf{T}_Q(0)$ on the element E_{Q-1}. This can be achieved by considering the induction hypothesis (12) for $\mathbf{T}_Q(u)$ and taking its value at $u = 0$,

$$\mathbf{T}_Q(0) = \mathbb{A}_Q^{\mathbf{T}}(0)\,\ddot{\boldsymbol{\kappa}}_Q + \mathbb{R}_Q^{\mathbf{T}}(0)\,\ddot{\mathbf{r}}_Q(0) + \mathbb{C}_Q^{\mathbf{T}}(0)\,\ddot{\boldsymbol{\theta}}_Q(0) + \mathbf{b}_Q^{\mathbf{T}}(0),$$

and by first replacing $\ddot{\boldsymbol{\kappa}}_Q$ with

$$\ddot{\boldsymbol{\kappa}}_Q = \left(\mathbb{A}_Q^{\mathbf{G}}\right)^{-1} \left(\mathbb{K}_Q\,(\boldsymbol{\kappa}_Q - \boldsymbol{\kappa}_Q^0) - \mathbb{R}_Q^{\mathbf{G}}\,\ddot{\mathbf{r}}_Q(0) - \mathbb{C}_Q^{\mathbf{G}}\,\ddot{\boldsymbol{\theta}}_Q(0) - \mathbf{b}_Q^{\mathbf{G}}\right) \tag{16}$$

using the induction hypothesis (10). Finally, we use the smoothness conditions (3b) of the kinematics for decomposing $\ddot{\mathbf{r}}_Q(0)$ and $\ddot{\boldsymbol{\theta}}_Q(0)$ on the element E_{Q-1},

$$\begin{cases}
\ddot{\mathbf{r}}_Q(0) = \ddot{\mathbf{r}}_{Q-1}(\ell_{Q-1})\\
\qquad = \ddot{\mathbf{r}}_{Q-1}(0) + \mathbb{A}_{Q-1}^{\ddot{\mathbf{r}}}(\ell_{Q-1})\,\ddot{\boldsymbol{\kappa}}_{Q-1}\\
\qquad\quad + \mathbb{C}_{Q-1}^{\ddot{\mathbf{r}}}(\ell_{Q-1})\,\ddot{\boldsymbol{\theta}}_{Q-1}(0) + \mathbf{b}_{Q-1}^{\ddot{\mathbf{r}}}(\ell_{Q-1})\\
\ddot{\boldsymbol{\theta}}_Q(0) = \ddot{\boldsymbol{\theta}}_{Q-1}(\ell_{Q-1})\\
\qquad = \ddot{\boldsymbol{\theta}}_{Q-1}(0) + \mathbb{A}_{Q-1}^{\ddot{\boldsymbol{\theta}}}(\ell_{Q-1})\,\ddot{\boldsymbol{\kappa}}_{Q-1} + \mathbf{b}_{Q-1}^{\ddot{\boldsymbol{\theta}}}(\ell_{Q-1}).
\end{cases} \tag{17}$$

We are thus able to express $\mathbf{T}_Q(0)$ as a function of $\ddot{\boldsymbol{\kappa}}_{Q-1}$, $\ddot{\mathbf{r}}_{Q-1}(0)$ and $\ddot{\boldsymbol{\theta}}_{Q-1}(0)$ only, and as a consequence to decompose $J_{i,Q-1}$ on the element E_{Q-1}.

Let us now decompose $I_{i,Q-1}$ on the element E_{Q-1}:

$$\begin{aligned}
I_{i,Q-1} &= \sum_{R>Q-1} \int_0^{\ell_R} \mathbf{T}_R(u) \cdot \left(\boldsymbol{\varphi}_{i,Q-1}(\ell_{Q-1}) \times \mathbf{n}_{0,R}(u)\right) \mathrm{d}u\\
&= \left(\sum_{R>Q-1} \int_0^{\ell_R} \mathbf{n}_{0,R}(u) \times \mathbf{T}_R(u)\,\mathrm{d}u\right) \cdot \boldsymbol{\varphi}_{i,Q-1}(\ell_{Q-1})\\
&= \mathbf{L}_{Q-1} \cdot \boldsymbol{\varphi}_{i,Q-1}(\ell_{Q-1})\\
&\text{with} \qquad \mathbf{L}_{Q-1} = \sum_{R>Q-1} \int_0^{\ell_R} \mathbf{n}_{0,R}(u) \times \mathbf{T}_R(u)\,\mathrm{d}u\\
&\qquad\qquad = \int_0^{\ell_Q} \mathbf{n}_{0,Q}(u) \times \mathbf{T}_Q(u)\,\mathrm{d}u + \mathbf{L}_Q.
\end{aligned}$$

The above recursive expression for $\mathbf{L}_Q$ proves that $\mathbf{L}_{Q-1}$ can be decomposed on the element E_Q using the induction hypothesis (12) for $\mathbf{T}_Q(u)$ and for $\mathbf{L}_Q$. We now need to show how to build the decomposition of $\mathbf{L}_{Q-1}$ on E_{Q-1}. The method is actually similar to the one employed for the decomposition of $J_{i,Q-1}$: the twist and curvatures acceleration $\ddot{\boldsymbol{\kappa}}_Q$ is replaced everywhere by its expression (16), then we use equations (17) for eliminating $\ddot{\mathbf{r}}_Q(0)$ and $\ddot{\boldsymbol{\theta}}_Q(0)$, and we finally express $\mathbf{L}_{Q-1}$ as a decomposition on E_{Q-1}. The decomposition of $I_{i,Q-1}$ on E_{Q-1} is then straightforward.

Finally, the decomposition for $G_{i,Q-1}$ on the element E_{Q-1} is obtained as the sum of the decompositions for $I_{i,Q-1}$ and for $J_{i,Q-1}$.