Towards Realistic Hair Animation Using Discrete Elastic Rods

Mila Grigorova

Master Thesis

MSc Computer Animation and Visual Effects
Bournemouth University
NCCA

September 2014

Abstract

When animating virtual humans, hairstyle is one of a most noticeable features that contribute to the authenticity of the character. Despite being a common task for visual effects industry, adequate simulation of hair still remains a challenging problem. Difficulties arise due to complex mechanical behaviour it exhibits, especially when considering the wide range of different hair styles. Although numerous method have been proposed in the literature there still is no well accepted model for hair. The objective of this thesis is set on accurately capturing the dynamics of the curly hair by investigating a physical model of a closely related phenomena, namely discrete elastic rods and its viability towards realistic hair animation.

1 Introduction

Realistic hair is an essential part in the creation of believable virtual characters for both visual effects industry and computer games. When animating virtual humans, one of the major challenges is to reproduce the hair movement in visually plausible and physically accurate manner. Although realistic hair simulation is a relatively old research topic in the field of computer graphics, finding a representation that provides efficient and accurate animation of hair motion remains an open challenge.

Human hair has inherently complex behaviour, being comprised of hundreds of thousands of thin and inextensible strands that interact with each other and with the body. This is further complicated when considering arbitrary shaped fibers. Each individual strand exhibits intricate nonlinear mechanical behaviour, which strongly correlates to its natural shape - straight, wavy or curly. In reality the structure, visual appearance and behaviour of hair is affected by multitude of factors. Some of them are intrinsic and vary from person to person such as strand cross-section, natural curvature and natural twist. Others depend on chemical, thermal or mechanical forces. This makes it very difficult, if not impossible, for any simulation scheme to account for all factors. Furthermore there is very little knowledge available regarding mutual hair interactions. Unlike other domains like solids or fluids where the underlying algorithms and techniques are governed by well established equations, there's still no single accepted model for hair [Ward et al. 07]. Due to its complex nature, algorithms that provide high visual fidelity pose strict requirements on simulation time steps. This, combined with the fact that a typical human has over 100,000 individual hair strands has an overwhelming effect on the performance and makes them unsuitable for interactive applications. Depending on the application area different simulation techniques focus on different goals - visual appearance, accuracy or performance.

The objective of this thesis is set on accurately capturing the dynamics of the curly hair by exploring an existing physical model of a closely related phenomena, namely elastic rods, and its applicability towards simulation of hair. As the hair's cross section is significantly smaller than its length it falls in the same category with other naturally curved filamentary structures such as ropes, plant tendrils, cables and even DNA . Surprisingly all of them show similar

mechanical behaviour strongly dependant on their intrinsic shape, though on widely different length scales [Reis et al. 14].

A standard physical framework that best describes the equilibria and ongoing deformations during motion for elastic rods in the continuous case is the Kirchhoff model. Kirchhoff theory of elastic rods assigns an elastic potential energy to each possible configuration of the rod in space which is expressed in terms of its curvature and twist. Equations governing the time evolution of the rod towards its equilibrium state are then derived from Lagrangian mechanics by feeding the Lagrangian equations of motion with the expression for the elastic energy [Audoly and Pomeau 10]. A number of recent papers aim to give a valid discretization of the underlying theory. Perhaps the most notable and popular amongst them is “Discrete Elastic Rods” by [Bergou et al. 08]. The authors present a discretization of the problem in an elegant manner using concepts from Discrete Differential Geometry and generalize their solution to naturally curved rods while accounting for anisotropic bending response. Although the paper is often cited in hair related literature, very little information could be found on actual implementations which confirm its viability to hair animation. To our knowledge only one such exists. More details about it could be found in a paper by [Kmoch et al. 09]. Furthermore it is surprising that no publicly available source code for the generalized case could be found online, although the method is mentioned in nearly every course on physically based animation. With this said, this thesis sets as a main goal to verify the paper’s relevance and adaptability towards realistic simulation of curly hair.

This document is structured as follows. Section 2 provides a cursory overview of the related literature. Section 3 gives brief overview of the theory behind Cosserat curves and Kirchhoff elastic rods. Section 4 presents the discretization model proposed by [Bergou et al. 08] and implemented for this project . Section 5 outlines of the scope of this project and goes into details regarding the key features and flaws of the current implementation of the proposed technique and the hair system itself and describes some deviations taken from the original solution. Section 6 concludes the subject with a discussion of the achieved results and applicability of the model to hair simulation. Furthermore it proposes future work directions.

2 Related work

Many computational models for simulation of individual hairs have been researched and developed for the past 25 years. They could be divided into two general categories: volume based and strand based. Methods based on continuum are more appropriate for smooth and straight hair while strand based approaches demonstrate better results for wavy and curly hair. As there are broad range of publications on this topic it is not possible to examine all of the techniques here in great detail. Therefore only a short description of the above mentioned methods is provided with a focus on most relevant work. For more extensive survey on the subject the reader is referred to [Ward et al. 07] and [Hadap et al. 07].

First attempts to achieve plausible animation of hair utilize mass- spring systems and were presented to the field by [Rosenblum et al. 91]. Mass- spring remains the most conventional model used in the industry due to its simple implementation and effectiveness. These models treat the hair explicitly as chain of hinges connected by stiff springs. Each particle in the chain has several degrees of freedom - positional and angular to account for bending. Inextensibility and bending rigidity are enforced by making the corresponding connective springs stiffer. One of the major drawbacks of such approach is its stability when dealing with stiff systems. Strong spring forces and enforcing inextensibility by limiting the stretching causes instabilities unless very small timesteps are used. A possible solution could be employing implicit integration as proposed by [Baraff and Witkin 98]. This however is often omitted for performance or other reasons. Many advances in mass-spring formulation were recently proposed. [Selle et al. 08] presented separate edge, bend,twist, and altitude springs which form implied tetrahedron, thus preventing volume collapse. [Petrovic et al. 12] incorporated a simplification of the elastic rod model by utilizing bending springs to reorient the material frame on a smoothed version of the curve describing the hair strand. In addition they used altitude springs to preserve its overall shape. A limitation to the spring model, however, remains fact that it can not properly account for twist.

Volume representation techniques were first introduced by [Hadap et al. 01]. Within this framework hair is treated as a volume of matter and simulated using continuum mechanics. A rough approximation of the motion is computed using

smoothed particles. Individual hair strands are still represented as chains of rigid links but motion is driven by the continuum particles. Other volumetric methods abandon the notion of individual strands even more markedly. In [Volino et al. 04], hair volume is modelled as a free-form lattice with nodes acting as simulated particles. Strands are attached to the lattice as viscoelastic springs.

As mentioned in the introduction, Kirchhoff theory of elastic rods provides valid and accurate physical model for thin, one- dimensional elastic structures such as hair. It was first introduced to the graphics community by [Pai 02] with application to the visualization of surgical wires. Later, this approach was extended by [Bertails et al. 06] to correctly reproduce the dynamics of helical elastic rods and was used to animate curled hairs. It was the earliest in this area of research and has provided the foundation for many of the papers proceeding it. Very recently [Bertails and Casati 13] presented super-clothoid model which introduces novel dedicated integration scheme to find the centerline deformation guaranteeing a stable simulation at the same time. All of these works use implicit representation of the rod by employing its curvature and twist as degrees of freedom in order to characterize the shape.

On the other hand several other methods use explicit representation, meaning that rod is approximated with a finite number of segments. Each segment has a material frame associated with it, which defines its orientation. Moreover this frame needs to track the tangential direction of the rod as it moves. This, however, can not be enforced in a straightforward manner when using the finite element definition. The suggested techniques differ in their parameterization and methodology used to update the segments' orientation and end-points. [Spillmann and Teschner 07] use quaternions to represent the material frame and utilize penalty forces in order to constrain it to follow the rod. The algorithm proposed by [Bergou et al. 08] was a breakthrough because of the elegant parametrization they used to describe the material frame at each segment. It is reconstructed from a reference frame called Bishop frame, which can be easily computed taking into account only the points on the centerline. This allows the orientation of the segments to automatically follow the deformation of the rod without the need to explicitly model torques acting on it. The method , however, requires smaller time steps as the stiffness of the rod and the number of approximating segments increases. To ensure proper stability at an acceptable computational cost, the authors recommend a fully implicit integration scheme based on Newton's method in a later paper [Bergou et al. 2010]. Lastly [Stam et al. 14], in their paper "Position- based Elastic Rods", provide formulation of the

problem within the efficient framework of position based dynamics where the elasticity is expressed as a system of constraints. As the authors outline, physical accuracy is not of particular concern to them. Rather, an efficient simulation is targeted capable of producing visually plausible results.

3 The physics behind Kirchhoff rods

A rod Γ is a deformable body whose one dimension (length) is significantly larger than the other two (cross section) i.e. a curve. It can be entirely described by the position of its centreline x(s) and an orthonormal material frame {t(s), m1 (s), m2 (s)} assigned for each point x(s). Here s denotes the arc- length parameter ranging from 0 to the total length L of the rod, x(s) is centreline position and m1,2 (s) are axes which define the cross section of the rod. The material frame is picked in such a way that t(s) is tangential to the curve at the particular point x(s) and its derivative with respect to s, t'(s) , represents the normal (curvature) vector of the curve. During deformation the material frame always stays orthonormal.

Energy

Kirchhoff's theory assigns an elastic energy $E(\Gamma)$ to any configuration of the rod in space. The energy is a scalar function which measures strain - the rate of change of the material frame. The material frame itself is expressed in its own local coordinates $\omega 1$, $\omega 2$, m:

$$\omega 1 = t'(s) \cdot m1(s) \, , \, \omega 2 = t'(s) \cdot m2(s) \, , \, m = m1'(s) \cdot m2(s)$$

Here $\omega 1$, $\omega 2$ represent the rod's curvature in local coordinates and measure bending of the material frame. Usually they are denoted together as a 2- vector $\omega = (\omega 1 \, , \, \omega 2 \,)^T$. The other term m represents the twist around the tangent direction t(s). The expression for energy is composed of a bending and twisting component and is given by the following formulas:

$$E(\Gamma) = E_{\text{bend}}(\Gamma) + E_{\text{twist}}(\Gamma)$$

$$E_{\text{bend}}(\Gamma) = \frac{1}{2} \int (\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})^T \overline{B} (\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}) \mathrm{d}s \, ,$$

$$E_{\text{twist}}(\Gamma) = \frac{1}{2}\int \beta m^2 \mathrm{d}s\ .$$

where $\overline{B}$ is a 2x2 symmetric matrix describing the rod's bending stiffness, β is a scalar describing the twisting stiffness and $\overline{\omega}$ denotes the curvature of the rod for its rest shape. Since inextensibility is assumed no stretching component is considered and length preservation is enforced as a separate post- integration step.

Bishop frame and parallel transport

The twist of the rod can be expressed as a scalar variable if we consider a twist free reference frame. Such frame is called Bishop frame and is denoted by {t(s), u(s), v(s)}. It is uniquely determined by fixing u(s0) and v(s0) at one of the ends of the rod. The evolution of the Bishop frame along the centreline can be expressed in terms of its Darboux vector Ω (s), for which the following holds true:

$$t' = \Omega \times t \qquad u' = \Omega \times u \qquad v' = \Omega \times v$$

By definition the Bishop frame is twist free meaning that m = u' * v = 0 , thus Ω has no tangential component and coincides with the curvature binormal kb = t × t'.

The Darboux vector of the Bishop frame is used to define the concept of parallel transport. Parallel transporting a vector v from one point of the centreline to another is achieved by integrating the equation v' = kb × v. This effectively corresponds to a rotation around the binormal kb. Parallel transport can be used to track the evolution the twist-free Bishop frame along the length of the centerline.

Twist representation

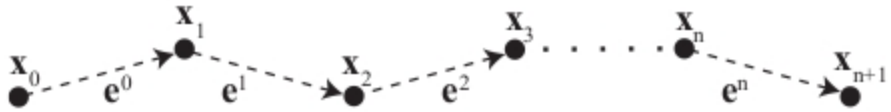
As said earlier we can can represent the twist m as scalar function θ(s) , which measures the angle (around the tangent t(s)) between the material frame and the reference Bishop frame. By doing so, the material axes m1 and m2 can be expressed in terms of the twist angle and the Bishop frame. This reduces the

number of parameters used to describe the rod at a particular point to 4 i.e. the position x(s) and the twist angle ϴ(s).

$$m1 = \cos(\theta)u + \sin(\theta)v,$$
$$m2 = -\sin(\theta)u + \cos(\theta)v$$

4 Discretization model

For a rod Γ(s) = {x(s), t(s), m1 (s), m2(s)}, its discrete description can be obtained by approximating it with n + 2 points x_0 , . . . , x_{n+1} connected by n + 1 segments e^0 , . . . , e^n .



Throughout this document, lower indices are used to denote quantities assigned to points and upper indices for those assigned to segments. A material frame {t^j , m1 , m2 } to each segment
e^j with the requirement to be adapted to the centreline, meaning $t^j = e^j / ||e^j||$.

As shown in [Bergou et al. 08] , a distinction must be made between quantities defined pointwise and those representing a value integrated over a domain. When an integrated quantity is associated with a point, its domain are the nearest halves of segments adjoined to the node. For node x_i , the domain has length l_i /2, where $l_i = ||e^{i-1}|| + ||e^i||$.

Bending energy

Discrete curvature binormal is an integrated quantity and together with the material curvature can be expressed by the following equations:

$$(\kappa \mathbf{b})_i = \frac{2\mathbf{e}^{i-1} \times \mathbf{e}^i}{|\bar{\mathbf{e}}^{i-1}||\bar{\mathbf{e}}^i| + \mathbf{e}^{i-1} \cdot \mathbf{e}^i} .$$

$$\boldsymbol{\omega}_i^j = \left((\kappa \mathbf{b})_i \cdot \mathbf{m}_2^j, -(\kappa \mathbf{b})_i \cdot \mathbf{m}_1^j \right)^T \quad \text{for} \quad j \in \{i-1, i\} .$$

Note that there are two curvature values ω_i^j defined for each point, one expressed for each adjoining segment. These allow to derive the following formula for bending energy:

$$E_{\text{bend}}(\Gamma) = \sum_{i=1}^{n} \frac{1}{2\bar{l}_i} \sum_{j=i-1}^{i} \left(\boldsymbol{\omega}_i^j - \overline{\boldsymbol{\omega}}_i^j\right)^T \overline{B}^j \left(\boldsymbol{\omega}_i^j - \overline{\boldsymbol{\omega}}_i^j\right) .$$

Bishop frame

In order to calculate the Bishop frame, within the discrete setting parallel transport is defined as a rotation P_i around the curvature binormal kb_i, or identity if $kb_i = 0$. We then fix the value of vector u0 (at the root of our rod) and parallel transport it along the centerline, thus obtaining $u^i = P_i (u^{i-1})$ and then set $v^i = t^i \times u^i$.

Twisting energy

With the Bishop frame defined, the material frame of segment j can be expressed using a rotation θ^j analogously to the continuous case:

$$m1 = \cos \theta^j u^j + \sin \theta^j v^j \quad m2 = -\sin \theta^j u^j + \cos \theta^j v^j$$

The same is done for the twisting energy:

$$E_{\text{twist}}(\Gamma) = \sum_{i=1}^{n} \beta \frac{(\theta^i - \theta^{i-1})^2}{\bar{l}_i} = \sum_{i=1}^{n} \frac{\beta m_i^2}{\bar{l}_i} ,$$

Material frame update

One of the important principles of the model described in [Bergou et al. 08] is the quasistatic treatment of twist. As the authors note the smaller the cross section of the simulated rod is the faster twist waves propagate. Thus twist can safely be removed as unknown from the equations.

When a simulation step finishes, before twist can be computed, the Bishop frame must be
updated. In general, after a simulation step ts , it can happen that $u^0 * t^0 \neq 0$. To realign the Bishop frame, we need to parallel transport it in time. This corresponds to rotation around axis defined by $t^0(t) \times t^0(t + \Delta t)$. The Bishop frame of the rest of the segments is then updated using normal parallel transport P_i .

With the Bishop frame updated, we can compute new twist, thereby updating the material
frame. The rod twists to minimize its internal energy E(Γ) only at unclamped segments. For clamped ends is value is already prescribed.

Equations of motion

The elastic force strives to minimize elastic energy. This is expressed in terms of the energy derivative :

$$F_{elastic}(x_i) = - dE(\Gamma) / dx_i$$

The total derivative of elastic energy takes into account both explicit dependence on centreline position and implicit dependence on it via the material frames. Therefore, to obtain an integrable formula, we must substitute into the total derivative:

$$-\frac{\mathrm{d}E(\Gamma)}{\mathrm{d}\mathbf{x}_i} = -\frac{\partial E(\Gamma)}{\partial \mathbf{x}_i} - \sum_{j=0}^{n} \frac{\partial E(\Gamma)}{\partial \theta^j} \frac{\partial \theta^j}{\partial \mathbf{x}_i} .$$

The material frame was obtained by requiring the derivative of E(Γ) with respect to θ^j to be 0 for unclamped points. Thus most of the terms in the summation above are 0. Remaining individual terms are given by the following expressions:

$$\frac{\partial E}{\partial \theta^n} = \frac{1}{\bar{l}_n} (\boldsymbol{\omega}_n^n)^T J \bar{B}^n (\boldsymbol{\omega}_n^n - \bar{\boldsymbol{\omega}}_n^n) + \frac{2\beta m_n}{\bar{l}_n} \quad \text{and}$$

$$\frac{\partial E}{\partial \mathbf{x}_i} = \sum_{k=1}^{n} \frac{1}{\bar{l}_k} \sum_{j=k-1}^{k} \left(\nabla_i \boldsymbol{\omega}_k^j\right)^T \bar{B}^j \left(\boldsymbol{\omega}_k^j - \bar{\boldsymbol{\omega}}_k^j\right) ,$$

$$\nabla_i \boldsymbol{\omega}_k^j = \begin{pmatrix} (\mathbf{m}_2^j)^T \\ -(\mathbf{m}_1^j)^T \end{pmatrix} \nabla_i (\kappa \mathbf{b})_k - \mathrm{J} \boldsymbol{\omega}_k^j (\nabla_i \Psi^j)^T .$$

$$\nabla_{i-1} (\kappa \mathbf{b})_i = \frac{2[\mathbf{e}^i] + (\kappa \mathbf{b}_i)(\mathbf{e}^i)^T}{|\bar{\mathbf{e}}^{i-1}||\bar{\mathbf{e}}^i| + \mathbf{e}^{i-1} \cdot \mathbf{e}^i} ,$$

$$\nabla_{i+1} (\kappa \mathbf{b})_i = \frac{2[\mathbf{e}^{i-1}] - (\kappa \mathbf{b}_i)(\mathbf{e}^{i-1})^T}{|\bar{\mathbf{e}}^{i-1}||\bar{\mathbf{e}}^i| + \mathbf{e}^{i-1} \cdot \mathbf{e}^i} ,$$

$$\nabla_i (\kappa \mathbf{b})_i = -(\nabla_{i-1} + \nabla_{i+1})(\kappa \mathbf{b})_i .$$

$$\nabla_i \Psi^j = \sum_{k=1}^{j} \nabla_i \psi_k \ .$$

$$\nabla_{i-1}\psi_i = \frac{(\kappa \mathbf{b})_i}{2|\bar{\mathbf{e}}^{i-1}|}, \qquad \nabla_{i+1}\psi_i = -\frac{(\kappa \mathbf{b})_i}{2|\bar{\mathbf{e}}^{i}|},$$

$$\nabla_i \psi_i = -(\nabla_{i-1} + \nabla_{i+1})\psi_i.$$

5 Implementation Details

This project implements the core concepts behind “Discrete Elastic Rods“ as described by [Bergou et al. 08] in an attempt to capture the complex mechanical behaviour of the individual hair filaments. The main idea was to incorporate the proposed solution as part of a more elaborate simulation system and put to a test its viability towards realistic hair animation. Some features mentioned in the paper, however, have been left out due to their irrelevance to the problem setup. One simplification that has been made is that just one- way coupling is considered as mainly the rod is affected by the motion of the body and not the other way around. As a consequence torque transfer on the rigid body described in the document has been omitted. Secondly, in contrast to the rods proposed in the paper, strands are allowed to self- intersect. Although in real life such scenario is virtually impossible this simplification doesn’t impair the model too much as the hair cross section is very small and in 3D setting self- collisions although possible are highly unlikely. Lastly, only isotropic bending response is considered meaning that the rod does not have a prefered bending direction. The anisotropic characteristics of the rod predominantly depend on its local cross-section which for hair can vary from oval to circular and is almost uniform along the strand length ([Bertails et al. 06]). Therefore, although this is not generally the case, for our model we assume that the filament has uniform circular cross section. Once the main algorithm is working, taking into consideration anisotropic bending should not be too cumbersome to add. This could be achieved by maintaining separate values for the bending stiffness along each of the two different material axes defining the cross sectional plane. Together they form a 2x2 diagonal matrix describing the rod’s flexural rigidity when no twisting is present. Thus to account for twist this matrix needs to be rotated by the twist angle for the corresponding segment.

The system simulates hair as an assembly of filaments. The hair volume is achieved with the use of interpolation. Only modest collection of individual leader strands are simulated and much larger number of follower strands are

generated from the leader ones for rendering purpose only. This allows to keep the simulation time manageable and in the same time capture the non-uniform behaviour of the hair volume. Because the typical hair consists of the large number of strands most of them will be in permanent contact with each other so self- collisions and stiction should also be considered in order to achieve appropriate level of realism. Different techniques exist for handling hair to hair contact. Some of them group the filaments into wisps and consider collisions only between wisps, while others accommodate methods from fluid dynamics such as SPH or FLIP. For this project an approach similar to the latter one is taken, where self- interactions are computed on a grid and added as external forces using a method described by [Petrovich et al. 05]. Its actual implementation was developed as a part of the CGIT assignment and has been modified and adapted to the current system. It should be noted, however, that any other method for handling the self contacts can to be used.

The implementation of the system utilizes three external libraries to achieve its goal - CML (Configurable Math Library) for general math calculations involving vectors, matrices and quaternions; dlib for numerical optimization and NGL to facilitate the work with the openGL API. The system has been designed with reuse in mind and attempts have been made to relieve it of as much specific data as possible. The configuration of the system is loaded at runtime from an external file containing the scene description in terms of geometry and specifying hair and simulation specific parameters. More extensive description of the available parameters can be found in the accompanying README file.

The following listing describes the algorithm and the main simulation loop:

(1) generate strands and precompute rest- state values
(2) compute quasi- static material frame based on Bishop frame from initial configuration
(3) for each simulation step:
(4) recalculate grid values based on strands' current configuration
(5) accumulate internal and external forces
(6) integrate equations of motion
(7) enforce inextensibility and handle collisions with body
(8) correct velocities
(9) update Bishop frame
(10) update quasi- static material frame

Hair generation

Hair generation process is carried out within the ***HairGenerator*** class. The hair can be attached to an arbitrary geometry. The strands' distribution and generation is controllable to some degree by the user. The filaments are placed only at the vertices of specified faces and their shape is fixed but configurable. Selection of faces is facilitated through Maya interface and python script is used to output the selected primitives. A huge disadvantage of placing the strands at the vertices of the mesh is that in this way the number of filaments and their location is coupled with its topology. An alternative and better approach would be to sample the surface of the geometry so that the number of generated strands and their distribution can be varied independently. A good solution to this problem could be utilizing Poisson sampling which distributes points in evenly spaced fashion.

As [Bertails et al. 06] state, the hair strand is entirely synthesized inside the follicle. Its mold pretty much determines the form of the strand, which is characterized by almost uniform cross section, natural curvature and twist. Thus, the fact that currently supported shapes are hard- coded in **HairGenerator** and can be either straight or helical with parameter controlling the radius and pitch of the helix, does not impair the visual output too much. This, nevertheless, limits the overall look of the hair volume. However, a flexible hairstyling tool falls outside the scope of this project.

Elastic rod initialization

Hair filaments are represented in the system by the **ElasticRod** class. Each elastic rod is initialized by its rest shape, a vector $\mathbf{u}^0$ fixing the Bishop frame for the rest shape at the root of the strand, the current configuration shape, mass and initial velocity for each point on the centerline as well as a set of boundary conditions denoting which points are clamped. Usually the hair strands are clamped only at their root position. However, the elastic rod was intended to be more general object and this is reflected in the code. Initial twist angles for each segment can also be suggested. These, however, will be overwritten by the minimization procedure in case energy minimization is switched on. After generating stress free rest shape for the strand, it is initialized using the same shape for the current configuration. During initialization curvature for the rest

shape is precomputed and stored using the formulas described in previous Section 4, after which the current state of the rod is updated.

Parallel transport and Bishop frame calculation

Updating the rod's state involves calculating a valid orientation for each segment and ensures that internal forces computed in the following simulation step are correct. It comprises of several steps. First, the fixed frame at the root segment is parallel transported in time, thus obtaining its orientation for the current configuration. Then the resultant frame is parallel transported along the the length of the curve thus obtaining the reference Bishop frame of the centerline.

Parallel transport is effectively a rotation around the curvature binormal $\mathbf{kb_i}$ with an angle $\mathbf{phi_i}$ defined by the two consequent segments $\mathbf{e^{i-1}}$ and $\mathbf{e^{i}}$:

$$\mathbf{\cos(phi_i) = dot(e^{i-1}, e^{i})}$$

Therefore it is most convenient this computation to be carried out with the use of quaternions. Moreover, the quaternion can constructed by extracting $\mathbf{\sin(phi_i / 2)}$ and $\mathbf{\cos(phi_i / 2)}$ from the length of the curvature binormal. As mentioned in the paper, the formula for $\mathbf{kb_i}$ produces:

$$\mathbf{||kb_i|| = 2 * \tan(phi_i / 2)}$$

thus speeding up the calculations a little bit. The above expression holds true only if the length of the corresponding edges is the same as their length for the rest shape, otherwise errors will occur in consequent frame rotations and thereby in the force calculations. Note that **tan** is a function that maps **(-pi / 2, pi / 2)** to **(-inf, inf)**, so for arbitrary real number an arbitrary angle would be extracted. As inextensibility of the rod is enforced explicitly through a system of constraints, in general deviations from the rest length should be small enough. However, a remark should be made that the above assumption can be violated if the constraint enforcement does not guarantee inextensibility at all cost as is the case with position based dynamics, described later in this section.

Energy minimization

As a final step of the update, energy minimization is performed to obtain the twist angle rotating the Bishop frame around the tangential direction of the rod and match it to the actual material frame. Within our discrete setting the curvature at a particular point $\mathbf{x_i}$ is dependant on the material frame at the adjacent segments $\mathbf{e^{i-1}}$ and $\mathbf{e^{i}}$ and in turn on twist angles $\theta^{\mathbf{i-1}}$ and $\theta^{\mathbf{i}}$ at those segments. In addition, the elastic potential energy affecting the deformation of the rod is expressed in terms of curvature and twist angle of all segments and establishes a non-linear relationship between them. For our model it is required that at any particular time the twist angles $\theta^{\mathbf{i}}$ minimize this elastic energy of the rod (see Section 4). Thus the current bending of the curve drives the values for the twist angles and removes them as unknowns for the next simulation step. Energy minimization is achieved with the use of dlib library. Dlib is a general purpose library with support for numeric algorithms and several optimization strategies for functions of higher dimensions - Newton, BFGS and conjugate gradient. The current implementation of **ElasticRod** provides the option the minimization method to be configured and changed at runtime as well as no minimization to be performed at all. The latter gives visually plausible results but reduces the stiffness of the rod and is not accurate (please refer to the video demonstrating differences in the deformations arising from using the different minimization methods).

Updating the rod's state is a routine that is also executed at the end of each simulation step and because of energy minimization is the bottleneck of the program.

Time integration

The equations of motion governing the rod's dynamic behaviour are expressed in terms of internal elastic force and external forces:

$$\mathbf{m_i * a_i = F_{elastic}(x_i) + F_{external}(x_i, v_i)}$$

Computation of $\mathbf{F_{elastic}(x_i)}$ is detailed in the previous section. $\mathbf{F_{external}(x_i, v_i)}$ is the net external force affecting the particle i. In our demonstration scenarios, we use gravity and friction against ambient air i.e.

$$\mathbf{F_{external}(x_i, v_i) = m_i * g - \nu * ||v_i|| * v_i}$$

where **g** is gravitational acceleration and ν is air drag coefficient. The equations of motion are integrated using the symplectic Euler method.

Internal and external forces acting on the rod are computed for all of its points at once. This, together with the fact that some additional state variables are maintained (edges, curvature binormals, material axes), improves the code readability which was given a preference over possible memory and performance gains. On the other hand given the large amount of computations involved in a single simulation step, it becomes infeasible to handle even moderate number of strands without the use of multithreading. Use of parallelism was also encouraged by the fact that the most intense part of calculations can be localized to the update routine of individual rods. Hence, it was possible the simulation to be easily distributed across multiple threads with the help of OpenMP (open multi processing).

Constraint enforcement

Stretching of the rod should also be considered when accounting for internal forces. So far the integration scheme involves no mechanism to maintain inextensibility, so length preservation must be ensured in some other way. As noted by [Bergou et al. 08] handling inextensibility by introducing stretching forces is undesirable as it leads to unnecessary stiff equations which pose restrictions on the time step which in turn impacts efficiency. Instead the authors deal with this in a separate post- integration step and enforce inextensibility and boundary conditions through satisfying a system of geometric constraints. Constraints are expressed as mathematical functions $\mathbf{C(x_1,x_2,...,x_n)}$ whose output is zero when the constraint is satisfied and non- zero otherwise. For instance the inextensibility constraint can be written as follows:

$$\mathbf{C(x_i,x_{i+1}) = ||x_{i+1} - x_i|| - d^i}$$

where **d** is the desired length of the corresponding segment. Another constraint applicable to our system is rigid- body coupling. In hair scenario it is used only to attach the root segment to the head. Thus the corresponding constraint takes the form:

$$\mathbf{C(x_0) = x_0 - p}$$

where **p** is the position of attachment. Finally collision response is also modeled as a constraint. Only collisions with ellipsoids are considered. The corresponding constraint can be expressed as:

$$C(x_i) = (x_{ix} - c_x)^2 / r_x^2 + (x_{iy} - c_y)^2 / r_y^2 + (x_{iz} - c_z)^2 / r_z^2 - 1$$

where **c** is the center of the ellipsoid.

There are a number of available approaches in the literature for maintaining constraints acting on a mechanical system. For example a method proposed by [Goldenthal et al. 07] and the one used in the paper by Bergou takes an unconstrained configuration and finds a “nearby” constrained one by iteratively solving a system of equations using Newton minimization. The word “nearby” is defined in terms of kinetic energy of the system.

Here, however, a different approach is taken and constraints are handled in Gauss-Seidel manner also known as position based dynamics(PBD) [Mueller et al. 06]. PBD is a technique that tries to resolve geometric constraints by manipulating the positions directly. The constraints have to be expressed in terms of positions. The main idea behind this method is that if a new eligible position for each point can be found which respects the constraints, then the unconstrained position and velocity of the point can be corrected explicitly. The positions are always modified directly, so the computations take into account previous corrections. The velocity is updated by assigning the difference between the new and the old positions just like when using Verlet integration scheme. This approach has the benefit of automatically correcting numerical errors accumulated during the integration step.

A point may be limited by multiple constraints and constraints may affect multiple points, therefore it is necessary to solve for all of the constraints at once. The problem here is that the constraints can generally take non- linear form (such as distance function above), so the new eligible positions can not be computed by solving a linear system of equations. Instead an iterative approach is taken where at each iteration the constraints are satisfied one by one in some order. When solving a constraint the relevant positions are directly manipulated and therefore the order in which the constraints are processed matters. In each iteration, particles may be moved closer or further away from each other and violate some of the previously solved constraints. This, however, is always

corrected in the following iteration, so the error becomes smaller and smaller with each loop.

Within the framework of PBD the attachment constraints can be satisfied by simply setting the position of the root to the attachment position:

$$\mathbf{x_0 = p}$$

For the distance constraint the two points are moved towards or away from each other proportional to their mass along the axis connecting them.

$$\mathbf{x_i = x_i + m_{i+1}/(m_i + m_{i+1}) * (||x_{i+1} - x_i|| - d^i) * normalize(x_{i+1} - x_i)}$$
$$\mathbf{x_{i+1} = x_{i+1} - m_i/(m_i + m_{i+1}) * (||x_{i+1} - x_i|| - d^i) * normalize(x_{i+1} - x_i)}$$

A possible optimization to relief the above calculations is to avoid computing the square root. To accomplish that the length of the corresponding segment is approximated with first order accurate Taylor expansion of sqrt function in the neighbourhood of $\mathbf{(d^i)^2}$. The reasoning behind it is that if the constraints are almost satisfied, the current length will not differ much from the rest one. This proved to be very useful as it removed some of underlying stiffness and made the simulation more stable.

$$\mathbf{l^i = 1 - 2 * (d^i)^2 / ((d^i)^2 + dot(x_{i+1} - x_i, x_{i+1} - x_i))}$$

In addition, clamped particles are thought to have infinite mass and thus their positions are never modified.

As mentioned before, collisions are handled as constraints. We approximate the colliding geometry with a number of ellipsoids. In this way a relatively accurate response can be generated at a very low computational cost. An arbitrary ellipsoid can be expressed as a 4x4 transformation matrix mapping the unit sphere to the ellipsoid itself. The actual collision is performed by projecting the offending points on closest point on its surface. In theory, this does not guarantee that all points will have valid positions after the projection step. Because of the multiple pbd iterations performed, however, penetrations go unnoticeable.

PBD produces good results even for small number of iterations (3 to 5), when the length of the segments is not too big. However, when testing with relatively long hair strands and small amount of elements per strand the stretch becomes

disturbingly apparent. To overcome this either the number of points or the number of iterations needs to be adjusted.

Hair to hair interactions

To achieve realistic results it is necessary to take into consideration interactions between individual fibers of the hair. Computing the hair-hair interactions on a particle level can be very time consuming considering the large amount of particles that are simulated. Searching for particles' neighbours is the bottleneck of such approach. To accelerate the computations a rough approximation of hair-hair interactions can be calculated on a grid. In their paper [Petrovic et al. 05] propose to construct a regular voxel grid which is rebuilt at the beginning of every simulation step. The grid should encompass the whole space where the hair can potentially move. Each voxel represents a volumetric sample of the hair volume and stores the average density and velocity of all the particles that happen to be within its range for the current frame. Using this information two forces are calculated for each particle - stiction force modelling the stiction between colliding strands and repulsion force modelling self collisions. When a particle is inserted into the grid, density with value one and its velocity are distributed amongst the eight neighbouring voxels using trilinear interpolation scheme. After all particles are considered each voxel holds a weighted average of the particles' velocities.

As already mentioned self stiction and self repulsion are accounted for as external forces. Hair stiction is computed by applying linear relaxation of the particle's current velocity $\mathbf{v_i}$ and an average velocity calculated based on the data held inside the grid $\mathbf{v_{i\_grid}}$. So a lookup for must be performed which in turn involves trilinear interpolation to calculate the velocity at the queried position. Obviously the voxel size determines how many particles contribute to the average velocity.

$$\mathbf{v_i = (1 - s_{stiction}) * v_i + s_{stiction} * v_{i\_grid} / d(x_i)}$$

Here $\mathbf{s_{stiction}}$ controls the strength of the stiction force and $\mathbf{d(x_i)}$ denotes interpolated the density at the particle's position.

To simulate hair repulsion, the pressure gradient $\mathbf{g}$ at the particle's position is approximated again from the grid data. The gradient is a vector pointing away from denser regions and is treated as a force. $\mathbf{s_{repulsion}}$ parameter controls the

strength of this force and acts as a constant multiplier.The pressure gradient is computed by performing six trilinear interpolations utilizing finite difference method.

$$\mathbf{g} = \begin{pmatrix} \partial d/\partial x \\ \partial d/\partial y \\ \partial d/\partial z \end{pmatrix} \approx \begin{pmatrix} (d(x-r,y,z) - d(x+r,y,z))/2r \\ (d(x,y-r,z) - d(x,y+r,z))/2r \\ (d(x,y,z-r) - d(x,y,z+r))/2r \end{pmatrix}$$

Here **d(x,y,z)** is the interpolated the density for the queried position **(x,y,z)**.

Hair geometry and visualization

The visualisation of the system has both a basic representation via openGL and exported output.The basic openGL preview is capable to display the simulated data at real time, though the simulation itself can achieve interactive rates only for small number of hairs (approximately 200), even with parallelization at hand. For preview only simulated strands are rendered.

Internally the simulated rod is represented as a sparse sequence of points. To obtain a smoother approximation of the centerline Catmull-Rom spline is used where particles' positions are treated as control points The reason behind this choice is that the resultant curve has couple of desired properties - it passes through all of the control points and closely approximates the available data and the tangent information is implied by the control points themselves. Strands are visualized as smooth tubes utilizing interpolation and the available material frame data. The generation of the necessary geometry is realized on the GPU, through the use of tessellation shader.

In addition, the simulation can be exported as a series of .obj files and post-processed and rendered using external 3D software package. For example interpolation of follower strands is handled inside SideFX Houdini. All the results shown in the accompanying videos were shaded and rendered also using Houdini.

6 Results and conclusion

The model has been tested with several different scenarios. For all of them a separate configuration file is provided and for most of them a demo animation is compiled. The scenes are setup so the model is validated against its capability to

handle different hair styles - straight, wavy, curly. As well as hair of different length was used to verify the model's independence on hair length. The obtained simulation results clearly show that elastic rods represent a valid solution to capturing the complex dynamics of curly hair. Rendered results demonstrate that visually pleasing animation can be achieved with even moderate amount of simulated data. Controlled and accurate experiments, however, have not been performed due to the short time span left for testing. Screenshots from the animations can be seen below.

Computational costs to obtain those results, however, were high. All tests were performed on an Intel Xeon E5-1650 @3.20GHz CPU running on all 12 cores with 31Gb RAM, NVIDIA Quadro K2000 2Gb GRAM machine. As mentioned previously, interactivity is hampered even for moderate number of strands (approximately 200) with relatively small number of particles up to 25. Thus the solution proves to be inappropriate when performance requirements are strict. Major bottleneck are energy minimization and handling self contacts. Moreover as [Kmoch et al. 09] note, simulating correct hair behaviour requires stiffnesses 2–3 orders of magnitude larger than scenarios considered by [Bergou et al. 08]. This poses restrictions on the simulation time step which for all tests performed here was set to 10ms. In addition to limitations imposed by higher stiffness it has been observed that increasing the number of elements also impacts stability. Thus possible production application at this point is arguable.

References

[Bergou et al. 08] Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., and Grinspun, E., 2008. Discrete Elastic Rods. ACM Transactions on Graphics, 27 (3), 63.

[Bergou et al. 10] Bergou, M., Audoly, B., Vouga, E., Wardetzky, M., and Grinspun, E. 2010. Discrete viscous threads. ACM Transactions on Graphics , Proc. ACM SIGGRAPH , 2010.

[Kmoch et al. 09] Kmoh, P., Bonanni, U., and Thalmann, N., 2009, Hair simulation model for real-time environments. In ACM SCA, 2009.

[Spillmann and Teschner 07] Spillmann, J., and Teschner, M., 2007, CoRdE: Cosserat rod
elements for the dynamic simulation of one-dimensional elastic objects. In ACM SCA, 2007.

[Stam et al. 14] Stam, J., Umetani, N., Schmidt, R., 2014. Position- based elastic rods. In Eurographics/ ACM SIGGRAPH SCA, 2014.

[Reis et al. 14] Reis, P., Audoly, B., Lazarus, A., Miller, J., 2014. Shapes of a Suspended Curly Hair. In PHYSICAL REVIEW LETTERS, 2014.

[Audoly and Pomeau 10] Audoly, B., and Pomeau, Y. 2010. Elasticity and Geometry: from hair curls to the nonlinear response of shells. Oxford University Press.

[Goldenthal et al. 07] Goldenthal, R., Harmon, D., Fattal , R., Bercovier, M., and Grinspun, E. 2007. Efficient simulation of inextensible cloth. In ACM TOG, 49.

[Baraff and Witkin 98] Baraff, D., and Witkin, A., “Large steps in cloth simulation,” Proc. of
ACM SIGGRAPH, pp. 43–54, 1998.

[Petrovic et al. 12] Iben, H., Meyer, M., Petrovic, L., Soares, O., Anderson, J., and Witkin, A., 2012 Artistic Simulation of curly hair. Pixar Animation Studios. SIGGRAPH, 2012.

[Bertails et al. 05] Bertails, F., Audoly, B., Querleux, B., Leroy, F., Lévêque, J. L., and Cani, M.P., 2005. Predicting Natural Hair Shapes by Solving the Statics of Flexible Rods . Eurographics Short Papers.

[Bertails et al. 06] Bertails, F., Audoly, B., Cani, M. P., Querleux, B., Leroy, F., & Lévêque, J. L., 2006. Super helices for predicting the dynamics of natural hair. ACM Transactions on Graphics, 25 (3), 11801187).

[Bertails and Casati 13] Bertails, F., and Casati, R., Super space clothoids. ACM TOG 32, 4 (2013).

[Pai 02] Pai, D. K., 2002. Strands: Interactive simulation of thin solids using cosserat models. In
Computer Graphics Forum, 21(3), 347352.

[Rosenblum et al. 91] Rosenblum, R. E., Carlson, W. E., & Tripp, E., 1991. Simulating the structure and dynamics of human hair: modelling, rendering and animation. The Journal of Visualization and Computer Animation, 2(4), 141148.

[Selle et al. 08] Selle, A., Lentine, M., & Fedkiw, R., 2008. A mass spring model for hair simulation. ACM Transactions on Graphics, 27(3), 64.

[Ward et al 07] Ward, K., Bertails, F., Kim, T. Y., Marschner, S. R., Cani, M. P., & Lin, M. C., 2007. A survey on hair modeling: Styling, simulation, and rendering.Visualization and Computer Graphics, IEEE Transactions on, 13 (2), 213234.

[Mueller et al. 06] Mueller, M., Heidelberger, B. , Hennix, M., and Ratcliff, J. , 2006. Position based dynamics. Workshop on Virtual Reality Interactions and Physical Simulation, 2006.

[Mueller et al. 12] Mueller, M., T. Y. Kim, and N. Chentanez 2012. Fast simulation of inextensible hair and fur.Workshop on Virtual Reality Interactions and Physical Simulation, 2012.

[Mueller et al. 08] Mueller, M. , Stam, J., Doug, J., Thurey, N., 2008. Real Time Physics. SIGGRAPH Course, 2008.

[Petrovic et al. 05] Petrovic, L. ,Henne, M. , and Anderson, J., 2005. Volumetric methods for simulation and rendering of hair. Technical report, Pixar, 2005.

[Henne 12] Henne, A., 2012. Simulation of hair and fur. University of Freiburg, 2012.

[Bridson et al. 02] Bridson, R. , Fedkiw, R., Anderson, J. , 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. Proceedings, SIGGRAPH, 2002.

[Hadap et al. 07] Hadap, S., Cani, M. P., Bertails F., Lin, M., Ward. K., Marschner, S. R., Kim, T. Y. and Alesic, Z., 2007. Strands and Hair: Modeling, Animation, and Rendering. ACM SIGGRAPH 2007 Courses.

[Hadap et al. 01] Hadap, S., and Magnenat-Thalmann, N. , 2001. Modeling dynamic hair as a continuum. Eurographics, 2001.

[Volino et al. 04] Volino, P., Thalmann, N., 2004. Animating complex hairstyles in real-time. In VRST '04 2004, ACM Press.

[Choe 05] Choe, B. , Choi, M., and Ko, H. S. , "Simulating complex hair with robust collision handling," in SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation. New York, NY, USA: ACM Press, 2005, pp. 153–160.

[Parent 12] Parent, R., 2012. Computer animation: algorithms and techniques. London : Morgan Kaufmann 2012