Examensarbete
LITH-ITN-MT-EX--07/029--SE

Super-Helices For Hair Modeling and Dynamics

Mattias Bergbom

2007-05-25

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Science and Technology
Linköpings universitet
SE-601 74 Norrköping, Sweden

Institutionen för teknik och naturvetenskap
Linköpings universitet
601 74 Norrköping

LITH-ITN-MT-EX--07/029--SE

Super-Helices For Hair Modeling and Dynamics

Examensarbete utfört i medieteknik
vid Linköpings Tekniska Högskola, Campus Norrköping

Mattias Bergbom

Handledare Prof. Ken Museth
Handledare Digital Domain and Linköping University
Handledare and Dr. Doug Roble
Handledare Digital Domain
Examinator Ken Museth

Norrköping 2007-05-25

Avdelning, Institution
Division, Department

Institutionen för teknik och naturvetenskap

Department of Science and Technology

Datum
Date

2007-05-25

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English

☐ _____

Rapporttyp
Report category

Examensarbete
☐ B-uppsats
☐ C-uppsats
☒ D-uppsats

☐ _____

ISBN

ISRN **LITH-ITN-MT-EX--07/029--SE**

Serietitel och serienummer
Title of series, numbering

ISSN

URL för elektronisk version

Titel
Title

Super-Helices For Hair Modeling and Dynamics

Författare
Author

Mattias Bergbom

Sammanfattning
Abstract

We present core components of a hair modeling and dynamics solution for the feature film industry. Recent research results in hair simulation are exploited as a dynamics model based on solving the Euler-Lagrange equations of motion for a discretized Cosserat curve is implemented in its entirety. Solutions to the dynamics equations are derived and a framework for symbolic integration is outlined. The resulting system is not unconditionally positive definite but requires balanced physical parameters in order to be solvable using a regular linear solver. Several implementation examples are presented, as well as a novel modeling technique based on non-linear optimization.

Nyckelord
Keyword

hair modeling dynamics super-helix

Abstract

We present core components of a hair modeling and dynamics solution for the feature film industry. Recent research results in hair simulation are exploited as a dynamics model based on solving the Euler-Lagrange equations of motion for a discretized Cosserat curve is implemented in its entirety. Solutions to the dynamics equations are derived and a framework for symbolic integration is outlined. The resulting system is not unconditionally positive definite but requires balanced physical parameters in order to be solvable using a regular linear solver. Several implementation examples are presented, as well as a novel modeling technique based on non-linear optimization.

Acknowledgements

None of this work would have been possible without the guidance, inspiration and patience of Professor Ken Museth, Digital Domain and Linköping University, who contributed with conceptual and mathematical knowledge and ideas far beyond his supervisory duties. Among many other things, the mathematics derivations presented in this thesis are in large part due to his original ideas. Thank you!

Dr. Doug Roble, Digital Domain, gave me the opportunity of a lifetime and made sure I did my very best to capture it. His knowledge, wit and enthusiasm paved the way for this work and made me go further in my efforts than I had ever hoped.

Nafees Bin Zafar, Digital Domain, provided the fundamental kinematics code as well as plenty of invaluable advise along the way. Also, without the input and encouragement from Serge Sretschinsky, Ross Kameny, Paul George Palop, Dan Patterson, Chris Harvey and Lauralea Otis, all with Digital Domain, this thesis would not have been nearly as interesting as it is today.

Finally, a great thank you to my parents (and main sponsors) Paula and Rolf, and my sister Sofia. For always being there, sharing my dreams and never once doubting I could make them come true.

Contents

Chapter 1

Introduction

Over the past decade the visual effects industry has seen most major companies develop in-house character animation pipelines. The need for proprietary pipelines stems from a need for flexibility and performance far beyond what any one commercial software can provide, coupled with economical and infrastructural considerations. Features such as "Spiderman", "King Kong" and the "Lord of the Rings" trilogy push these systems to the limit and beyond, as the main characters often are entirely CG and the audience - being human - is inherently good at noticing abnormal appearance in other humanoids.

An important part of visualizing characters on screen is hair, a complex task in many aspects; modeling, grooming, simulating, interacting with and finally rendering hair are all topics subject to extensive research. This thesis was written while implementing some of the latest advances in hair modeling and dynamics at Digital Domain (DD), one of the most prominent visual effects companies in the industry. While the thesis doesn't attempt to solve all the problems related to putting hair on the big screen, it does present a few key components of a hair pipeline, as well as outline its potential use in a large scale character pipeline.

1.1 Purpose and aim

The purpose of this thesis project has been to implement and examine the Super-Helix model for hair modeling and dynamics simulation, as proposed by Bertails *et al.* in [3]. This in order to support a decision regarding whether to adopt the model into the character pipeline at Digital Domain, and to facilitate a possible adoption. The aim has been perform the math derivations behind and implementation of the Super-Helix model for dynamics, to examine the modeling aspects, and to present a software implementation that is usable yet extensible, along with a discussion of its qualitative

properties.

1.2 Outline

The thesis is organized as follows. Initially, chapter 2 summarizes existing research in the field. Chapter 3 presents a brief overview over the existing hair/character pipeline at Digital Domain, and sketches the parts of the new pipeline pertaining to hair modeling and rendering. Then focus turns to the actual kinematics and dynamics as chapters 4 and 5 present the full Super-Helix derivations and implementation, respectively. Finally, chapter 6 sums up the results and discusses the qualitative properties of the model.

Chapter 2

Previous work

We divide previous work into three categories: styling, dynamics and rendering. All three categories have seen extensive research over the years, where each publication usually deals with a fairly narrow special case. For brevity, we only present the relevant parts of the styling and dynamics categories here, as the scope of this thesis largely excludes rendering.

2.1 Surveys

For an introduction to the topic of hair in computer graphics, Ward *et al.* [26] present an extensive overview, as well as provide several detailed studies and key insights. Its contributors include some of the most prominent persons currently active in the field, many of whom have at least one publication included in the survey. While this author would have prefered a second glance at the field by an alternate source, no such publication was readily available to the public at the time this thesis was written.

2.2 Styling

A large body of work has been published on modeling and grooming hair in creative ways, all with the general objective to increase productivity and creative control while maintaining realism. The user interface is given attention by Malik [15] who uses a stylus pen interface with abilities to implant, cut and style hair in an intuitive way. The result is a low learning threshold and very rapid results (10-20 minutes for a basic hairstyle [15]), at the cost of realism and detail [26]. Kim and Neumann present thin shell volumes for governing hair growth [14] and a multi-resolution hierarchical modeling technique with the capability to copy-paste detail from one area to another [13]. Other methods include fluid-flow styling by Hadap and Magnenat-Thalmann, vector fields by Yu [30] and photography-based hair modeling

by e.g. Wei *et al.* [28].

Some physically based modeling efforts have also been made, but they remain on the experimental level [26]. The user sacrifices direct control to achieve a higher degree of realism in the actual underlying physics and visual appearance.

2.3 Dynamics

Hair dynamics simulation is a very hard task indeed, as hair strands are highly anisotropic in their flexible behavior and the interactions between hairs are complicated by an anisotropic surface microstructure and triboelectric properties [26].

A multitude of approaches to hair dynamics simulation have been presented through the years. One can divide the most interesting advances into three categories: strand/wisp, multi-resolution and full continuum models. These will be described in more detail below.

It is important to consider what application the simulation will have. In computer games and other realtime applications realism must always be weighted against performance, while for feature film production some speed usually can be traded for increased accuracy and level of control. Creating matching hair grooms for several different dynamics models is generally undesirable, due to the vast time consumption involved in any one setup.

2.3.1 Strands and wisps

The perhaps most intuitive way to model hair is to start by looking at the individual hair strands and try to mimic their behaviour using some method from the field of mechanical engineering. To achieve large-scale effects the simulated hairs are relatively sparsely populated and intermediate hairs are statically interpolated over the surface as a post-step to simulation, see e.g. [3] and [6]. Another approach is to use the strand as a "skeleton curve", governing the deformation of a *wisp* [19], a virtual volume envelope which is filled with hairs at render time. While the two approaches are sufficiently similar to be presented in the same section, an important difference is that the wisps explicitly enclose their hairs in isolated clumps, while interpolative methods don't necessarily attribute an interpolated hair to any one guide hair. In broad terms this gives the latter an edge in representing straight and wavy hair with a frequent of forming and unforming of clumps, while the former is better suited to handle curly hair where clumps are more static.

The *mass-spring model*, as originally introduced by Rosenblum *et al.* in 1991 [21], is perhaps the simplest and most straightforward approach to

strand/wisp based simulation. The hair is represented as number of point masses, connected by springs, and all interactions are modeled as forces and constraints affecting the masses. The mass-spring model is an example of a *nodal* model, meaning any global response to a local interaction, such as a collision, needs to be advanced through the nodes at a speed bounded by the timestep size. This is fine for isotropic, "springy" objects, but in order to simulate e.g. the longitudinal rigidity of a hair strand the response forces have to be ramped up to extents that cause the simulation to explode unless the timestep size is reduced several orders of magnitude. Thus improving the performance of the large-scale dynamics - which are of the most interest visually - is overshadowed by a nearly infinitesimal timestep restriction in order to prevent springiness. This is refered to as "stiffness" in the equations and is a major obstacle when designing dynamics models.

To address the issues in mass-spring models, methods such as *one-dimensional projective equations* by Anjyo *et al.* [11] and *rigid multi-body serial chains* due to Hadap and Magnenat-Thalmann [9] and improved upon by Chang *et al.* [6], have been proposed. Both maintain the nodal based approach but restrict the degrees of freedom to only include bending and torsion. This precludes stretching already at the model stage. However, the former does not account for torsional rigidity and needs special care to deal with punctual forces. The latter, being formulated in reduced coordinates, suffers from difficulties in implementing hard constraints using an implicit integration scheme and the only proposed alternative solution is "non-trivial" [26].

Moreover, neither of the above models properly handles deformations of curly hair and non-linear buckling and bending-twisting discontinuity effects [3]. Bertails *et al.* [3] present an approach to simulating the dynamics of Cosserat curves, as originally introduced to the CG community for statics simulation by Pai [16]. Their model - called the *Super-Helix* due to its composition of elements of constant helicity - accurately handles a multitude of non-linear effects previously never addressed in computer graphics. While external forces enter the equations in a very elegant way, being formulated in reduced coordinates - much like the multi-body serial chains - the model does not handle hard constraints well. In fact, implementing hard constraints in the Super-Helix model can be "tricky" [26].

Recently, hybrid schemes have been devised by e.g. Choe *et al.* [7] that combine the strengths of several different nodal methods and use implicit integration, global constrained dynamics and other recent advances gathered from the neighboring fields of flexible bodies and cloth. While they alleviate many of the problems mentioned, they are still battling several issues inherent in their underlying methods such as inefficient inter-hair collision detection [7] and some amounts of stretching [26].

2.3.2 Multi-resolution methods

An important feature of natural hair is the clumping of hair strands due to frictional forces, and the ongoing forming and dissolution of clumps. While the wisp models essentially mimic this behaviour in a static way, only recently have mutual hair interactions been exploited mainly for performance reasons. Ward *et al.* [27] use discrete levels of detail (LOD) based on viewing distance, hair motion and visibility to reduce computation times intelligently. In their *Adaptive Wisp Tree* (AWT) method [5], Bertails *et al.* model mutual interactions by employing a tree structure for a hierarchical subdivision of wisps. While rendering at the finest level of every branch, simulation is adaptively done at any level, depending on the local complexity of motion. It helps mainly to increase the efficiency and stability of previous wisp-based simulation approaches, but [7] suggests it might also help simulate wisps with unclear boundaries and strands shifting between wisps, an inherent problem in that approach.

2.3.3 Continuum models

An entirely different approach to simulating hair - presented by Hadap and Magnenat-Thalmann in [9] - is to view it as a continuum, and model its motion using fluid dynamics. This technique is based on Smooth Particle Hydrodynamics (SPH) and handles complex hair interactions and collisions nicely. However, apart from an inability to handle certain clumping and other small-scale effects it is prohibitively expensive computationally-wise; computation times of several minutes per frame for 10,000 hair strands have been reported [26].

Chapter 3

Hair pipeline at DD

At the core of the hair pipeline at Digital Domain is Autodesk's Maya Hair, first featured in Maya version 6 and subsequently improved upon in versions 7 and 8. Maya Hair is a strands/wisps based package with a mass-spring model at its core. It takes NURBS curves as input geometry and uses Maya's Paint Effects system to render "clump hairs" post simulation, according to user-provided parameters. Consequently, most modeling techniques in the pipeline aim at achieving a "target" or "rest" configuration of the guide hairs, upon which dynamics simulations are run. Pixar's RenderMan is used to render the final shots so the Paint Effects settings are exported to a custom RenderMan shader, which uses that information and the guide hair curves to generate the final result.

While having proven sufficient in films such as "Aeon Flux", "My Super-Ex Girlfriend" and "King Kong", the current pipeline has considerable room for improvement. The general consensus among artists and supervisors was that although the modules work in theory, any larger scale hair rigging projects will inevitably suffer from the lack of mature grooming tools, limited or inaccurate visual feedback and unreliable simulation results. As in any visual effects pipeline various tricks and cheats are employed to adapt the available tools to show-specific tasks, but with sparse reuse of previous results and a steep learning curve for new hires, efficiency could be improved considerably by replacing and/or complementing certain key components of the pipeline [23] [18] [17].

3.1 Styling

The artists often work with different components of the hairdo separately, for example dealing with the two sides of a parting separately or creating the bangs disjoint from the scalp hair. In fact, oftentimes more than one hair dynamics system will be employed in the same hairdo, as different lengths

of hair tend to require different parameters to the model, and to improve manageability in general. This unfortunately precludes any physical interaction between the different pieces of the hairdo.

With a very limited set of grooming tools available in Maya Hair, and with the added requirement of having to bake and render anything in RenderMan for visual feedback, artists employ various tricks to achieve the wanted results using the least amount of time. An informal survey revealed which techniques are most commonly used, most of which have been devised solely to try to work around the limitations of the pipeline.

Figure 3.1: The hair rig for the character "Aeon Flux" from the feature film with the same name. The "gravity" based modeling technique was employed, altogether taking about two weeks to finish. In the end a radically simpler model with a more specific rest setting was used, since the Maya Hair system had problems handling the extreme motions and geometry collisions in the sequence. Image courtesy of Dan Patterson and Digital Domain.

Gravity based modeling is motivated by the lack of coherence among adjacent guide hairs in Maya Hair. Since deformations made to one guide hair are not distributed to its neighbors, grooming a dense set of guide hair by means of direct nodal manipulation is unfeasible. By employing the dynamics solver, the artist gains a larger-scale control over the flow of the hair. For example, applying a constant gravitational force and tilting the model's head in some direction makes the hair fall in that direction, giving the effect of a very crude comb. The technique is usually applied to subsets

of the hairdo at the time by locking the other parts, and is particularly useful when creating parted, straight haircuts, such as the page of the main character in the production "Aeon Flux" (see Figure 3.1). However, getting the hair to fall just right is extremely tedious and many iterations usually have to be performed. [18]

Textures for placement and shape are another attempt to remedy the lack of interpolation over neighboring guide hairs. For example, by painting directions with a broad, blurry brush, one can achieve a kind of interpolation. While this seems like a sufficient solution, it is still quite tedious and, again, results are not entirely visible until after recomputing the cache, simulating and rendering. Furthermore, Maya Hair does not adhere exactly to placement textures, meaning stray hairs regularly have to be removed in compositing.[17]

Collision objects are commonly used as primitive proxy force objects to help shape the hair when gravity is activated. For instance, a properly placed collision sphere can prevent hair from obstructing the face of a character, guiding the hair to fall as though it was under the influence of internal stiffness and curl, where none such actually is present.[23]

Hair helmets are based on the thin shell volumes technique by Kim and Neumann [14]. The artist models a surface object that defines the volume that the hair should fill. The hair is then grown into the volume from its roots, bending and flowing along the surface according to some direction vector. Usually the NURBS surface isoparms (i.e. the lines on the surface representing points of the same parameter in the NURBS equation) are used to control the direction although various methods are feasible. This technique has worked really well in some productions but takes a great deal of care in setting up, with low reusability between different setups [23].

3.2 Dynamics

The only hair dynamics solution currently in use at Digital Domain is Maya Hair. Although a pinnacle in the history of computer animation, interviews with professionals with extensive experience in the field ([18], [23], [12]) reveal that the package was never fully finished as a product, and suffers from implementation flaws severe enough to adversely affect or even prohibit many uses in feature film production.

First, due to the stiffness problem discussed in section 2.3, any mass-spring based dynamics model will have a certain amount of inherent flexion, leading to stretching of hair strands. While increasing the stiffness of the springs remedies the stretching, pushing it to the extents necessary to simulate hair

causes stiff timestep constraints and severe performance problems [26] [6]. Maya Hair not only stretches, but the lack of hard constraints is also known to cause dislodging of the hair roots from the underlying surface [23]. This is especially the case in the presence of large temporal derivatives (e.g. large changes in position between frames), a very common scenario when working with data from motion capture or tracking devices, which in turn are ubiquitous in feature film production. Some fixes have been provided by Autodesk, although it remains unclear whether these in fact succeed to address the issue properly [12]. In any event this general weakness in nodal models is significant enough to warrant extensive research efforts to find other approaches [6].

Furthermore, colliding Maya Hair against arbitrary geometry imposes a severe restriction on the timestep size. The exact cause of the problem is unknown, but an attempt to remedy this was made by Autodesk by providing two specific collision objects - a sphere and a box - rather than using ordinary scene geometry. These can be bound in the usual way to transformations of scene geometry and used to fill other geometric shapes, but approximating shapes from spheres and boxes always introduces the risk of hairs getting caught in creases and interfaces, due to numerical imprecision and/or imprecise volume fitting [12]. The consensus among artists and developers was that some further form of collision geometry needs to be added, preferably from a modular, in-house collision engine which handles both detection of and response to collisions. As Maya Hair provides no facilities for incorporating a custom collision solution, this was one of the motivations to develop a proprietary hair system.

3.3 Rendering

At the time of writing, Digital Domain relies heavily on Pixar's RenderMan for rendering 3D elements, hair being one of them. This means that the Paint Effects settings done in Maya need to be exported and somehow emulated with RenderMan shaders. Using a custom RenderMan plugin named *Dogbeard*, the sparse set of guide hairs output from the simulation are interpolated to form the tens or hundreds of thousands of hairs that make up a complete hairdo. The interpolation is essentially linear, but numerous parameters are exposed, such as curliness, frizz, clumping etc. While there exist scripts to bake Paint Effects settings to Dogbeard, there is no 1-to-1 mapping between the two, so consequently the artists need to bake the entire hairdo and render it in RenderMan to see the impact of changes made at the modeling stage. As some recent (non-human) characters have been populated by tens of thousands of guide hairs, the resulting tediousness of creating previews has severely deteriorated the artists' productivity [17]. Implementing new, more accurate, preview tools in Maya was seen as a possible extension to the results of this thesis.

While Dogbeard has proven successful in various productions - and not only strictly for hair; feathers and grass are other examples - it suffers from one major drawback: interpolated hairs are static and do not in any way participate in the dynamics simulation. This has several repercaussions: First, whatever detailed geometry wedges between two guide hairs (a common example would be the starched collar of a shirt) will interpenetrate with an arbitrary amount of the interpolated hairs between those two guide hairs. Second, the hairs themselves will perform no interpenetration checks; while this can be somewhat remedied using heuristics for damping and friction, side effects include loss of volume in dense hair as no volume preservation is enforced, potentially causing the hair to assume an overly flimsic and otherwise unphysical look.

Also, as all communication within Dogbeard uses plain NURBS curves,

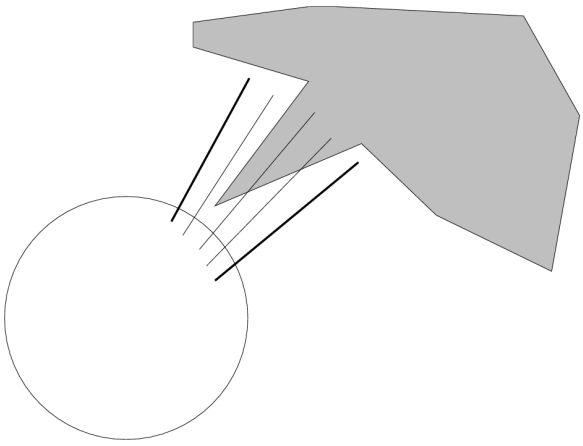

Figure 3.2: A sharp object wedges between two guide hairs, causing the interpolated hairs to penetrate it. This can have highly undesirable, visible effects such as hair penetrating the collar of a shirt.

no normals are defined at the modeling stage and thus must be conjured up at render time. While this might not seem relevant to hair of perfectly circular cross-section, most hair does in fact display a certain ellipticity [3] and thus requires the rendering and dynamic behaviour to match. Also, other applications than plain hair (again, grass perfectly examplifies this) might very well require well defined normals to shade properly, if at all. This exposed another, different, kind of need which was also taken into account when settling for a model.

Chapter 4

The Super-Helix model

A key objective of this thesis was to explore the Super-Helix model by Bertails *et al.*. As shown in chapters 2 and 3 the interest in a new dynamics solver in general and the Super-Helix model in particular was motivated by several factors, including the issues with the current hair pipeline at DD and the very promising results published in [3], [2] and [26]. While the strengths and weaknesses of most other models are fairly well-established in the community, putting effort toward investigating the very recent Super-Helices and verifying their usability seems particularly worthwhile.

4.1 Preliminaries

The Super-Helix concept is fundamentally based on the Elastica theory developed by Euler and Bernoulli, and later on generalized by brothers Cosserat. The *Cosserat curve* is a special case of the broader Cosserat theory for the elasticity of shells, rods and points.

Pai introduced the Cosserat concept into the computer graphics community with the STRANDS paper [16], dealing with the statics of surgical wire. Bertails *et al.* extended Pai's work by presenting a new formulation of the Kirchhoff equations for elastic rods [4], introducing collision response into the statics equations. The results look visually interesting but Bertails *et al.* later showed that the Kirchhoff equations are very stiff and thus expensive to solve numerically [3]. They also do not handle the dynamics of the hair, meaning they are of questionable value in a character animation scenario. Instead Bertails *et al.* propose a so called *global* approach, inspired by the Lagrangian deformable models promoted by Baraff and Witkin [1]. Baraff and Witkin describe how the integration of large-scale effects due to local interactions in any nodal (i.e. "non-global") model for flexible bodies requires timesteps that are "potentially disastrous" for performance [1]. They also show that their derived expressions minimize the deviation between the

resulting motion and the motion of an ideal continuum body.

The Super-Helix model addresses several of the issues that plague previous models. First and foremost, the degrees of freedom of the model are exclusively in the curvatures, meaning non-stretching is inherently and strictly enforced with no constraints on the generalized coordinates. Furthermore, since its degrees of freedom naturally comply with helical shapes, it deals with curly hair strands with unprecedented accuracy, not just individually but also collectively. Also, being a global model, it handles buckling and other effects in real hair that are due to its longitudinal stiffness and, again, not easily emulated by nodal models [2].

The terms "nodal" and "global" are used here to distinguish between models that deal with interactions locally (and thus suffer from stiffness as discussed in section 2.3) and models that handle all interactions globally. As we will see, the Super-Helix model belongs to the latter, even though one could argue that it in a sense still is nodal.

4.1.1 The Cosserat curve

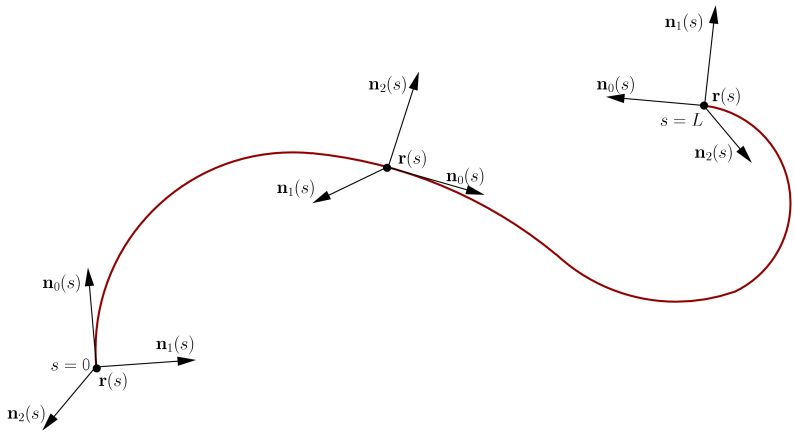

Figure 4.1: The Cosserat curve is described by a centerline $\mathbf{r}(s)$ and a local material frame $(\mathbf{n}_i(s))_{i=0,1,2}$ that is defined at every point along the curve.

The fundamental component of the Super-Helix model is the Cosserat curve, which consists of a centerline $\mathbf{r}(s)$ and an orthogonal *material frame* $(\mathbf{n}_i(s))_{i=0,1,2}$ (see Figure 4.1), with $\mathbf{n}_0(s)$ denoting the tangent $\mathbf{r}'(s)$ and $\mathbf{n}_1(s)$ and $\mathbf{n}_2(s)$ denoting the normal and binormal, respectively. For each infinitesimal step δs along the curve, this local space, defined by the position $\mathbf{r}(s)$ and the material frame, is affected by an affine transformation (i.e. translation and rotation) according to three local *curvatures* $(\kappa_i(s))_{i=0,1,2}$. Here κ_0 denotes

torsion (i.e. the rotation of the normals $\mathbf{n}_1(s)$ and $\mathbf{n}_2(s)$ around the tangent $\mathbf{n}_0(s)$) and κ_1 and κ_2 (rather ambiguously) denote the *curvature*, i.e. the local change of the normal $\mathbf{n}_0(s)$ in the $\mathbf{n}_1(s)$ and $\mathbf{n}_2(s)$ directions. An important effect of this parametrization in space is that the curve is C^1-smooth, meaning the centerline and material frames are continuous in s.

The local parameterization of the Cosserat curve is similar to that of the *Frenet-Serret* formulas, in that a material frame is attached to each point along the curve, and that curvatures are defined based on the local change of this frame. The difference is that the Frenet-Serret model only has *one* curvature (in the $\mathbf{n}_1$ direction) and one torsion. This means it can describe any curve *locally*, while it is more limited than the Cosserat model given a fixed material frame (see Figure 4.2). In fact, and as is further discussed in section 4.2.9, the flexibility of the Cosserat model comes at the price of some amount of dependency between the curvature parameters, which affects the conditioning of the system. We will however see how the flexibility given a fixed material frame is crucial to the Super-Helix model.

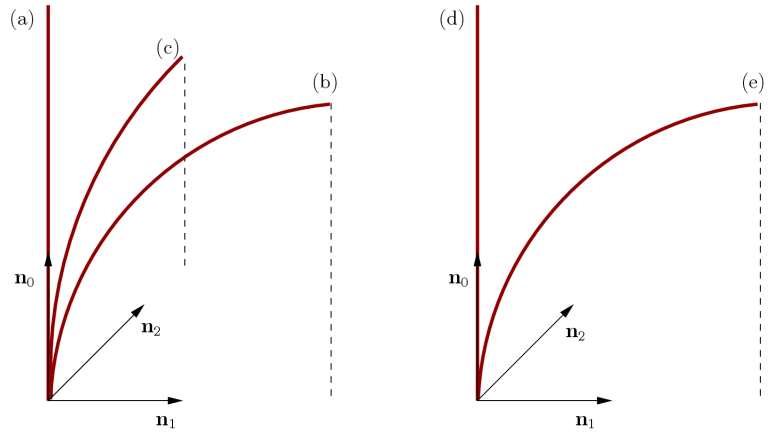

Figure 4.2: A Cosserat curve (a) and a Frenet-Serret curve (b) have similar formulations but the Cosserat curve has a greater flexibility given a particular material frame. While the Cosserat curve is free to bend in both the $\mathbf{n}_1$ (b) and $\mathbf{n}_2$ (c) directions the Frenet-Serret curve is limited to bending in the $\mathbf{n}_1$ (e) direction.

4.1.2 Super-Helix reconstruction

Instead of posing equations for the Cosserat model of continuous curvature and integrating numerically in s, Bertails *et al.* perform an initial discretization by assuming the curve is divided into N helical segments S_Q of constant curvatures $(q_{i,Q})_{i=0,1,2,Q\in[0..N]}$. The curve becomes a collection perfectly helical segments; hence the name "Super-Helix". We write the

discretized curvature function

$$\kappa_i(s, \mathbf{q}) = \sum_{Q=0}^{N} \chi_Q(s) q_{i,Q} \tag{4.1}$$

where $\chi_Q(s)$ is the Dirac function

$$\chi_Q(s) = \begin{cases} 1 \text{ if } s \in S_Q \\ 0 \text{ else} \end{cases} \tag{4.2}$$

which simply tells which three curvature values to use for a specific interval $s \in S_Q$. In order to maintain the C^1 continuity of the Cosserat curve, Bertails *et al.* constrain the initial position and orientation of each segment to be the final position and orientation of the preceeding segment - or the initial values of the entire SH in the case of the root segment. Given these intervals of constant curvature, reconstructing the Super-Helix amounts to piecewise integration of what resembles standard helix equations. The C^1 continuity constraint governs the initial values of each segment, as visible in Figure 4.3.

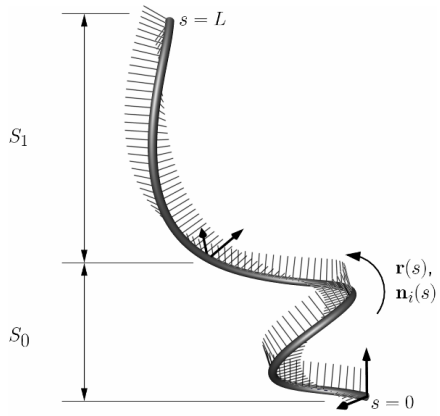

Figure 4.3: A two-segment Super-Helix. The reconstruction of the centerline (grey) and material frame (black) is visible throughout the strand. Notice how the C^1 continuity constraint causes the normals and bi-normals to seamlessly transition from one segment to the next.

The Super-Helix concept is described in some detail in [2] and [3], but both publications avoid to point out a lot of the peculiarities that are of tremendous importance when actually implementing the Super-Helix model. In order to complement this, we start by defining all of the involved variables, paying extra attention to indices and dependencies. The initial position and

coordinate frame can be written

$$\mathbf{n}^0_{i,L}(t) = \mathbf{g}_i(t) \quad \text{for } i = 0, 1, 2 \tag{4.3}$$

$$\mathbf{r}^0_L(t) = \mathbf{h}(t) \tag{4.4}$$

where $\mathbf{g}$ and $\mathbf{h}$ are inputs from e.g. keyframe animation or interactive manipulation. The notation gets a bit hairy here (no pun intended), as the superscript 0 denotes the 0:th segment, the subscript L denotes the left - i.e. initial - point of the segment, and i as usual enumerates the material frame axes.

The *Darboux vector* is an essential part of the equations. For each position s along the Super-Helix it is defined as the curvature vector $(\kappa_i(s))_{i=0,1,2}$ transformed into the orthogonal space spanned by the local material frame $(\mathbf{n}_i)_{i=0,1,2}$, such that

$$\begin{aligned}\mathbf{\Omega}(s, \mathbf{q}, t) = \mathbf{n}^Q_0(s, \mathbf{q}, t)\kappa_0(s, \mathbf{q}) + \mathbf{n}^Q_1(s, \mathbf{q}, t)\kappa_1(s, \mathbf{q}) \\ + \mathbf{n}^Q_2(s, \mathbf{q}, t)\kappa_2(s, \mathbf{q}),\end{aligned} \tag{4.5}$$

with the superscript Q indicating variables pertaining to segment S_Q. By the properties of the Cosserat curve, the Darboux vector describes the rate of change of the material frame in s:

$$\mathbf{n}'_i(s, \mathbf{q}, t) = \mathbf{\Omega}(s, \mathbf{q}, t) \times \mathbf{n}_i(s, \mathbf{q}, t) \quad \text{for } i = 0, 1, 2 \tag{4.6}$$

Since the Super-Helix has piecewise constant curvatures, and the coordinate frame is orthogonal, one can show that the Darboux vector is piecewise constant as well [3]:

$$\mathbf{\Omega}' = \sum_i \kappa'_i \mathbf{n}_i + \mathbf{\Omega} \times \mathbf{\Omega} = \mathbf{0} \tag{4.7}$$

We now present some definitions that are utilized throughout the Super-Helix reconstruction. Note the insertion of the initial values, and how they

impose explicit time dependencies in several places.

Darboux vector in initial position:

$$\mathbf{\Omega}_0(\mathbf{q},t) = \mathbf{\Omega}(0,\mathbf{q},t) = \mathbf{g}_0(t)\kappa_0(0,t) + \mathbf{g}_1(t)\kappa_1(0,t) + \mathbf{g}_2(t)\kappa_2(0,t) \tag{4.8}$$

Length of Darboux vector:

(explicit t dependency vanishes due to orthogonality of $\mathbf{n}_i$)

$$\Omega(s,\mathbf{q}) = |\mathbf{\Omega}(s,\mathbf{q},t)| \tag{4.9}$$

Length of Darboux vector in segment S_Q:

$$\Omega_Q(\mathbf{q}) = \Omega(s,\mathbf{q}), s_Q \in S_Q$$

Normalized Darboux vector:

$$\omega(s,\mathbf{q},t) = \mathbf{\Omega}(s,\mathbf{q},t)/\Omega(s,\mathbf{q}) \tag{4.10}$$

Shorthand for normalized Darboux vector in S_Q:

$$\omega_Q(\mathbf{q},t) = \omega(s,\mathbf{q},t), s \in S_Q \tag{4.11}$$

Initial material frame for segment S_Q:

$$\mathbf{n}^Q_{i,L}(\mathbf{q},t) = \mathbf{n}^Q_i(s^Q_L,\mathbf{q},t) \tag{4.12}$$

To take the next step towards a complete reconstruction of the Super-Helix, the parallel and perpendicular projections of the i:th coordinate axis span a plane in which lies the s derivative of that coordinate axis:

Parallel projection of coordinate axis onto normalized Darboux vector (in segment S_Q):

$$\mathbf{n}^{Q\parallel}_{i,L}(\mathbf{q},t) = \Big(\mathbf{n}^Q_{i,L}(\mathbf{q},t)\cdot\omega_Q(\mathbf{q},t)\Big)\omega_Q(\mathbf{q},t) \tag{4.13}$$

Perpendicular projection:

$$\mathbf{n}^{Q\perp}_{i,L}(\mathbf{q},t) = \mathbf{n}^Q_{i,L}(\mathbf{q},t) - \mathbf{n}^{Q\parallel}_{i,L}(\mathbf{q},t) \tag{4.14}$$

For element S_0; note the time dependence coming from 4.3:

$$\mathbf{n}^{Q_0\parallel}_{i,L}(\mathbf{q},t) = \Big(\mathbf{n}^{Q_0}_{i,L}(t)\cdot\omega_{Q_0}(\mathbf{q},t)\Big)\omega_{Q_0}(\mathbf{q},t) \tag{4.15}$$

$$\mathbf{n}^{Q_0\perp}_{i,L}(\mathbf{q},t) = \mathbf{n}^{Q_0}_{i,L}(t) - \mathbf{n}^{Q_0\parallel}_{i,L}(\mathbf{q},t) \tag{4.16}$$

Now we can write the reconstruction of the material frame along a segment S_Q. Note how the dependence on the entire preceeding part of the Super-Helix enters recursively through the $\mathbf{n}^{Q\parallel}_{i,L}$ and $\mathbf{n}^{Q\perp}_{i,L}$ terms:

$$\begin{aligned}\mathbf{n}^Q_i(s,\mathbf{q},t) = {}& \mathbf{n}^{Q\parallel}_{i,L}(\mathbf{q},t) + \mathbf{n}^{Q\perp}_{i,L}(\mathbf{q},t)\cos(\Omega_Q(\mathbf{q})(s-s^Q_L)) \\ & + \omega_Q(\mathbf{q},t)\times\mathbf{n}^{Q\perp}_{0,L}(\mathbf{q},t)\sin(\Omega_Q(\mathbf{q})(s-s^Q_L)).\end{aligned} \tag{4.17}$$

As 4.6 suggests, the change in the material frame is closely related to the Darboux vector. Indeed 4.17 describes how the coordinate axes of the material frame rotate about the Darboux vector with angular velocity Ω_Q. In

the case of $\kappa_0 \neq 0$, $\kappa_1 = 0$, $\kappa_2 = 0$, the Darboux vector aligns with the tangent $\mathbf{n}_0$ (see eqn. 4.5), which in effect is not subject to any rotation.

$$\begin{aligned}
&\textit{Coordinate axis } i \textit{ at start of segment } S_Q,\\
&\textit{as a function of } S_{Q-1}\textit{:}\\
\mathbf{n}^Q_{i,L}(\mathbf{q},t) &= \mathbf{n}^{Q-1}_{i,R}(\mathbf{q},t) = \mathbf{n}^{Q-1}_{i}(\ell_{Q-1},\mathbf{q},t)\\
&= \mathbf{n}^{Q-1\parallel}_{i,L}(\mathbf{q},t) + \mathbf{n}^{Q-1\perp}_{i,L}(\mathbf{q},t)\cos(\Omega_{Q-1}(\mathbf{q})(\ell_{Q-1}))\\
&+ \omega_{Q-1}(\mathbf{q},t)\times\mathbf{n}^{Q-1\perp}_{0,L}(\mathbf{q},t)\sin(\Omega_{Q-1}(\mathbf{q})(\ell_{Q-1})) \qquad (4.18)\\
&\textit{Coordinate axis } i \textit{ at start of segment } S_{Q_1},\\
&\textit{as a function of } S_{Q_0}\textit{:}\\
\mathbf{n}^{Q_1}_{i,L}(\mathbf{q},t) &= \mathbf{n}^{Q_0}_{i,R}(\mathbf{q},t) = \mathbf{n}^{Q_0}_{i}(\ell_{Q_0},\mathbf{q},t)\\
&= \mathbf{n}^{Q_0\parallel}_{i,L}(\mathbf{q},t) + \mathbf{n}^{Q_0\perp}_{i,L}(\mathbf{q},t)\cos(\Omega_{0}(\mathbf{q})(\ell_{Q_0}))\\
&+ \omega_{Q_0}(\mathbf{q},t)\times\mathbf{n}^{Q_0\perp}_{0,L}(\mathbf{q},t)\sin(\Omega_{0}(\mathbf{q})(\ell_{Q_0})). \qquad (4.19)
\end{aligned}$$

As stated in equations 4.3 and 4.8, $\mathbf{\Omega}_0$ depends explicitly on time, which together with 4.15 and 4.16 shows the explicit time dependence of $\mathbf{n}^{Q_0\perp}_{i,L}$ and $\mathbf{n}^{Q_0\parallel}_{i,L}$. Equations 4.17-4.19 then show how this *explicit time dependence is advanced throughout the reconstruction* of the Super-Helix.

Now we turn to the reconstruction of the centerline $\mathbf{r}^{SH}$. Since $\mathbf{n}_0 = \mathbf{r}'$, integrating $\mathbf{n}_0$ symbolically gives us the centerline $\mathbf{r}^{SH}(s,\mathbf{q},t)$ as an initial value $\mathbf{r}^Q_L(\mathbf{q},t)$ (entering as an integration constant) plus a function $\mathbf{N}_Q(s,\mathbf{q},t)$ that depends on the initial frame of reference and curvature.

$$\begin{aligned}
\mathbf{r}^{SH}(s,\mathbf{q},t) &= \mathbf{r}^Q_L(\mathbf{q},t) + \mathbf{n}^{Q\parallel}_{0,L}(\mathbf{q},t)(s-s^Q_L) + \mathbf{n}^{Q\perp}_{0,L}(\mathbf{q},t)\frac{\sin(\Omega_Q(\mathbf{q})(s-s^Q_L))}{\Omega_Q(\mathbf{q})}\\
&+ \omega_Q(\mathbf{q},t)\times\mathbf{n}^{Q\perp}_{0,L}(\mathbf{q},t)\frac{1-\cos(\Omega_Q(\mathbf{q})(s-s^Q_L))}{\Omega_Q(\mathbf{q})}. \qquad (4.20)
\end{aligned}$$

This is all for an arbitrary segment S_Q. Noting that $\mathbf{r}^Q_L = \mathbf{r}^{Q-1}_R$, we see how the final value of the previous segment enters as the initial value of this segment, so

$$\begin{aligned}
\mathbf{r}^{SH}(s,\mathbf{q},t) &= \mathbf{r}^{Q-1}_R(\mathbf{q},t)\\
&+ \mathbf{n}^{Q\parallel}_{0,L}(\mathbf{q},t)(s-s^Q_L) + \mathbf{n}^{Q\perp}_{0,L}(\mathbf{q},t)\frac{\sin(\Omega_Q(\mathbf{q})(s-s^Q_L))}{\Omega_Q(\mathbf{q})}\\
&+ \omega_Q(\mathbf{q},t)\times\mathbf{n}^{Q\perp}_{0,L}(\mathbf{q},t)\frac{1-\cos(\Omega_Q(\mathbf{q})(s-s^Q_L))}{\Omega_Q(\mathbf{q})}.
\end{aligned} \qquad (4.21)$$

Repeating this leads to the recursion

$$
\begin{aligned}
&= \mathbf{r}_L^{Q-1}(\mathbf{q},t) \\
&+ \mathbf{n}_{0,L}^{Q-1\parallel}(\mathbf{q},t)(\ell_{Q-1}) + \mathbf{n}_{0,L}^{Q-1\perp}(\mathbf{q},t)\frac{\sin(\Omega_{Q-1}(\mathbf{q})(\ell_{Q-1}))}{\Omega_{Q-1}(\mathbf{q})} \\
&+ \omega_{Q-1}(\mathbf{q},t) \times \mathbf{n}_{0,L}^{Q-1\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_{Q-1}(\mathbf{q})(\ell_{Q-1}))}{\Omega_{Q-1}(\mathbf{q})} \\
&+ \mathbf{n}_{0,L}^{Q\parallel}(\mathbf{q},t)(s-s_L^Q) + \mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{\sin(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})} \\
&+ \omega_Q(\mathbf{q},t) \times \mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})} \\
&\dots
\end{aligned}
$$

and finally

$$
\begin{aligned}
&= \mathbf{r}_L^{Q_0}(t) \\
&+ \sum_{\hat{Q}=0}^{Q-1}\left[\mathbf{n}_{0,L}^{\hat{Q}\parallel}(\mathbf{q},t)(\ell_{\hat{Q}}) + \mathbf{n}_{0,L}^{\hat{Q}\perp}(\mathbf{q},t)\frac{\sin(\Omega_{\hat{Q}(\mathbf{q})}(\ell_{\hat{Q}}))}{\Omega_{\hat{Q}}(\mathbf{q})}\right. \\
&\left.+ \omega_{\hat{Q}}(\mathbf{q},t) \times \mathbf{n}_{0,L}^{\hat{Q}\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_{\hat{Q}}(\mathbf{q})(\ell_{\hat{Q}}))}{\Omega_{\hat{Q}}(\mathbf{q})}\right] \\
&+ \mathbf{n}_{0,L}^{Q\parallel}(\mathbf{q},t)(s-s_L^Q) + \mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{\sin(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})} \\
&+ \omega_Q(\mathbf{q},t) \times \mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})}
\end{aligned}
\tag{4.22}
$$

4.2 Dynamics

To avoid the stiffness problems in previous hair dynamics models, Bertails *et al.* turn to *Lagrangian mechanics*. Using the curvatures of the kinematics model as the *generalized coordinates* in a Lagrangian formulation, they exploit the discretization of the Cosserat curve to describe a potentially very complex shape using a relatively sparse set of parameters. The resulting system is of size $3N$ where N is the number of segments in the Super-Helix, usually in the range of 5 to 15. This reduction of dimensionality is an attractive property of Lagrangian models. The key feat however is the way the Super-Helix reconstruction process described above models the inextensibility of hair, without imposing any constraints on the generalized coordinates (i.e. curvatures). By using the Super-Helix model in a Lagrangian formulation Bertails *et al.* thus effectively remove the main cause of stiffness in previous - nodal - models [3].

The basic concept of working with Lagrangian models (as described in [29]) is to associate masses with the generalized coordinates and affect them with generalized forces, which cause a displacement of the masses from which one can derive kinetic energy. By relating the internal, "elastic" energy with the deviation from some rest state, one obtains a set of equations which, for example, can be used in an energy minimization scenario.

4.2.1 Euler-Lagrange equations

The Euler-Lagrange equations of motion are a result of *Hamilton's principle*, which bounds the energy of the path between two states in the configuration space of a generalized system [24]. We omit the full derivation here, but the general result can be written

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0 \tag{4.23}$$

where i indicates that several coordinates might be involved. Bertails *et al.* write the Lagrangian $L = T - U$, where T is kinetic energy and U is potential energy, and include a heuristic damping term D. Adding a generalized force $\mathbf{Q}$, one obtains

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = \mathbf{Q}_i \tag{4.24}$$

Expanding this equation by again introducing the index Q to indicate that the involved coordinates relate to segment S_Q, and writing out the generalized force $\mathbf{Q}$ as the external force $\mathbf{F}$ translated to curvature coordinates (see section 4.2.2),

$$\frac{\mathrm{d}}{dt}\left(\frac{\partial T}{\partial \dot{q}_{iQ}}\right) - \frac{\partial T}{\partial q_{iQ}} + \frac{\partial U}{\partial q_{iQ}} + \frac{\partial D}{\partial \dot{q}_{iQ}} = \int_0^L \mathbf{J}_{iQ}(s, \mathbf{q}, t) \cdot \mathbf{F}(s, t) \mathrm{d}s. \tag{4.25}$$

Solving this equation for the change in $\mathbf{q}$ gives us the path of least energy for the evolution of the Super-Helix in time, which is indeed the true path according to Hamilton's principle [24]. In the specific case of the Super-Helix, the kinetic energy T is defined in [2] and [3] as

$$T(\mathbf{q}, \dot{\mathbf{q}}, t) = \frac{1}{2}\int_0^L \rho S \left(\dot{\mathbf{r}}^{SH}(s, \mathbf{q}, t)\right)^2 \mathrm{d}s \tag{4.26}$$

where ρ is the mass density and S is the elliptical cross-section area, both assumed constant. The internal potential energy is written

$$U(\mathbf{q}, t) = \frac{1}{2}\int_0^L \sum_{i=0}^{2} (EI)_i \left(\kappa_i^{SH}(s, \mathbf{q}) - \kappa_i^n(s)\right)^2 \mathrm{d}s \tag{4.27}$$

where the κ_i^n are the *natural curvatures*, in other words the rest state of the Super-Helix. Assuming an isotropic material, E is Young's modulus (set to approximately $4 \cdot 10^9$Pa) and I_i are the moments of inertia, given in [2] as

$$I_1 = \frac{\pi}{4} a_1 a_2^3$$
$$I_2 = \frac{\pi}{4} a_1^3 a_2$$

with the torsional component defined slightly differently as

$$(EI)_0 = \mu J$$
$$\mu = \frac{E}{2(1+\sigma)}$$
$$J = \pi \frac{a_1^3 a_2^3}{a_1^2 + a_2^2}$$

where σ is Poisson's ratio (set to 0.48). The heuristic model for dissipation potential D is written

$$D(\mathbf{q}, \dot{\mathbf{q}}, t) = \frac{1}{2} \int_0^L \gamma \sum_{i=0}^{2} \left(\dot{\kappa}_i^{SH}(s, \mathbf{q}) \right)^2 \mathrm{d}s \tag{4.28}$$

in [3] and

$$D(\mathbf{q}, \dot{\mathbf{q}}, t) = \frac{1}{2} \int_0^L \mu \sum_{i=0}^{2} (EI)_i \left(\dot{\kappa}_i^{SH}(s, \mathbf{q}) \right)^2 \mathrm{d}s \tag{4.29}$$

in [2], and essentially models energy dissipation due to inter-hair frictional forces. The reasoning behind the difference in definitions is that γ approximately encompasses the $\mu(EI)_i$ factors, but it turns out (see section 4.2.8) that the choice of model has a considerable effect on the system. We chose to use 4.29.

4.2.2 Differentiation

As will become apparent, in order to solve the system differentiation needs to be performed in two different variables, namely curvature and time. We start with the gradient $\mathbf{J}_{iQ} = \frac{\partial \mathbf{r}^{SH}(s,\mathbf{q},t)}{\partial q_{iQ}}$ which describes the change in position at a point along the Super-Helix related to the change in the curvature q_{iQ}. In this context it can intuitively be thought of as a transform from Cartesian space to curvature space, in particular in the generalized force case, as visible in e.g. equation 4.25. Using the Super-Helix reconstruction equation 4.22,

it can be written

$$\begin{aligned}\mathbf{J}_{iQ}(s,\mathbf{q},t) = \underbrace{\frac{\partial \mathbf{r}_L^{Q_0}(t)}{\partial q_{iQ}}}_{=0} + \sum_{P=Q}^{\hat{Q}-1} \frac{\partial}{\partial q_{iQ}} \Bigg[& \mathbf{n}_{0,L}^{P\parallel}(\mathbf{q},t)(\ell_P) + \mathbf{n}_{0,L}^{P\perp}(\mathbf{q},t)\frac{\sin(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})} \\ & + \omega_P(\mathbf{q},t) \times \mathbf{n}_{0,L}^{P\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\Bigg] \\ & + \frac{\partial}{\partial q_{iQ}}\Bigg(\mathbf{n}_{0,L}^{\hat{Q}\parallel}(\mathbf{q},t)(s-s_L^{\hat{Q}}) + \mathbf{n}_{0,L}^{\hat{Q}\perp}(\mathbf{q},t)\frac{\sin(\Omega_{\hat{Q}}(\mathbf{q})(s-s_L^{\hat{Q}}))}{\Omega_{\hat{Q}}(\mathbf{q})} \\ & + \omega_{\hat{Q}}(\mathbf{q},t) \times \mathbf{n}_{0,L}^{\hat{Q}\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_{\hat{Q}}(\mathbf{q})(s-s_L^{\hat{Q}}))}{\Omega_{\hat{Q}}(\mathbf{q})}\Bigg) \end{aligned} \tag{4.30}$$

where $\hat{Q}$ is the segment that s currently is in, i.e. $s \in \hat{Q}$. This relation to the segmentation means that the gradient assumes one of three considerably different shapes depending on where s and $\hat{Q}$ are (see Figure 4.4):

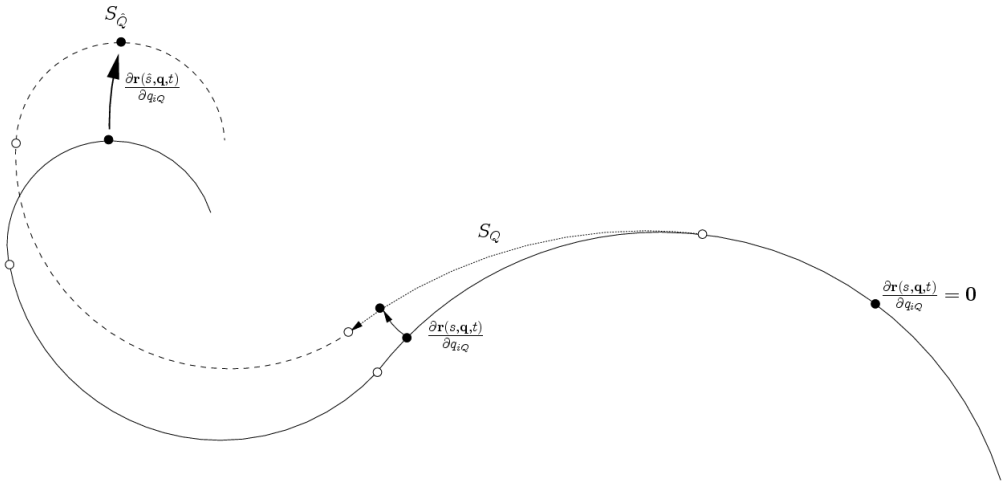

Figure 4.4: Change in curvature in segment Q affects all subsequent segments $\hat{Q}$ but no preceeding ones. The segment S_Q acts in a flexible manner while subsequent segments are rigid, altogether giving the symbolic solution to the gradient three distinct appearances depending on where it is evaluated.

for $\hat{Q} < Q$, the gradient is with respect to curvatures of a succeeding segment, so $\mathbf{J}_{iQ}(s,\mathbf{q},t) = \mathbf{0}$

for $\hat{Q} = Q$ the gradient is with respect to the curvatures of the current segment, so 4.30 simplifies to:

$$\mathbf{J}_{iQ}(s,\mathbf{q},t) = \frac{\partial}{\partial q_{iQ}}\Bigg(\mathbf{n}_{0,L}^{Q\parallel}(\mathbf{q},t)(s-s_L^Q) + \mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{\sin(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})} + \omega_Q(\mathbf{q},t)\times\mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})}\Bigg) \tag{4.31}$$

where Ω_Q is a function of q_{iQ}.

for $\hat{Q} > Q$, the gradient is with respect to curvatures of a preceeding segment, so derivatives of the initial values of the current segment must be included:

$$\begin{aligned}\tilde{\mathbf{J}}_{iQ}(s,\mathbf{q},t) = \frac{\partial}{\partial q_{iQ}}\Bigg[&\mathbf{n}_{0,L}^{Q\parallel}(\mathbf{q},t)(\ell_Q) + \mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{\sin(\Omega_Q(\mathbf{q})(\ell)_Q))}{\Omega_Q(\mathbf{q})} \\ &+ \omega_Q(\mathbf{q},t)\times\mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\frac{1-\cos(\Omega_Q(\mathbf{q})(\ell_Q))}{\Omega_Q(\mathbf{q})}\Bigg] \\ &+ \sum_{P=Q+1}^{\hat{Q}-1}\Bigg[\frac{\partial\mathbf{n}_{0,L}^{P\parallel}(\mathbf{q},t)}{\partial q_{iQ}}(\ell_P) + \frac{\partial\mathbf{n}_{0,L}^{P\perp}(\mathbf{q},t)}{\partial q_{iQ}}\frac{\sin(\Omega_P(\ell_P))}{\Omega_P} \\ &+ \frac{\partial}{\partial q_{iQ}}\left(\omega_P(\mathbf{q},t)\times\mathbf{n}_{0,L}^{P\perp}(\mathbf{q},t)\right)\frac{1-\cos(\Omega_P(\ell_P))}{\Omega_P}\Bigg] \\ &+ \frac{\partial\mathbf{n}_{0,L}^{\hat{Q}\parallel}(\mathbf{q},t)}{\partial q_{iQ}}(s-s_L^{\hat{Q}}) + \frac{\partial\mathbf{n}_{0,L}^{\hat{Q}\perp}(\mathbf{q},t)}{\partial q_{iQ}}\frac{\sin(\Omega_{\hat{Q}}(s-s_L^{\hat{Q}}))}{\Omega_{\hat{Q}}} \\ &+ \frac{\partial}{\partial q_{iQ}}\left(\omega_{\hat{Q}}(\mathbf{q},t)\times\mathbf{n}_{0,L}^{\hat{Q}\perp}(\mathbf{q},t)\right)\frac{1-\cos(\Omega_{\hat{Q}}(s-s_L^{\hat{Q}}))}{\Omega_{\hat{Q}}}\end{aligned} \tag{4.32}$$

Note that only the first bracket holds Ω that depend on the current Q. Thus, all succeeding terms are more easily differentiated. Grasping the concept of these gradients - how information flows from segment to segment - is essential to understanding the Super-Helix dynamics.

Now we differentiate with respect to time t, assuming $s \in S_{\hat{Q}}$. Since $\mathbf{r}^{SH} = \mathbf{r}^{SH}(s,\mathbf{q}(t),t)$, we obtain one explicit and one implicit dependence on t. Thus the Chain Rule gives

$$\dot{\mathbf{r}}^{SH}(s,\mathbf{q}(t),t) = \frac{\mathrm{d}\mathbf{r}^{SH}(t)}{\mathrm{d}t} + \sum_{Q=0}^{N}\frac{\partial\mathbf{r}^{SH}(\mathbf{q}(t))}{\partial q_{iQ}}\frac{\mathrm{d}q_{iQ}}{\mathrm{d}t}$$

$$
\begin{aligned}
&= \frac{\mathrm{d}\mathbf{r}^0_{0,L}(t)}{\mathrm{d}t} + \frac{\mathrm{d}\mathbf{n}^{\hat{Q}\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}(s - s^{\hat{Q}}_L) + \frac{\mathrm{d}\mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}\frac{\sin(\Omega_{\hat{Q}}(\mathbf{q})(s - s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\
&+ \frac{\mathrm{d}}{\mathrm{d}t}\left(\omega_{\hat{Q}}(\mathbf{q},t) \times \mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)\right)\frac{1 - \cos(\Omega_{\hat{Q}}(\mathbf{q})(s - s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\
&+ \sum_{P=0}^{\hat{Q}-1}\left[\frac{\mathrm{d}\mathbf{n}^{P\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}(\ell_P) + \frac{\mathrm{d}\mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}\frac{\sin(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right. \\
&\left.+ \frac{\mathrm{d}}{\mathrm{d}t}\left(\omega_P(\mathbf{q},t) \times \mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)\right)\frac{1 - \cos(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right] \\
&+ \sum_{P=0}^{\hat{Q}}\sum_{i=0}^{2}\underbrace{\frac{\partial \mathbf{r}^{SH}(s,\mathbf{q},t)}{\partial q_{iP}}}_{\mathbf{J}_{iP}(s,\mathbf{q},t)}\frac{\mathrm{d}q_{iP}}{\mathrm{d}t}.
\end{aligned} \tag{4.33}
$$

Here we define $\dot{\mathbf{r}}^{SH}_{exp}$ as the "explicit" part of $\dot{\mathbf{r}}^{SH}$,

$$
\begin{aligned}
\dot{\mathbf{r}}^{SH}_{exp}(s,\mathbf{q}(t),t) = &\frac{\mathrm{d}\mathbf{r}^0_{0,L}(t)}{\mathrm{d}t} + \frac{\mathrm{d}\mathbf{n}^{\hat{Q}\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}(s - s^{\hat{Q}}_L) + \frac{\mathrm{d}\mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}\frac{\sin(\Omega_{\hat{Q}}(\mathbf{q})(s - s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\
&+ \frac{\mathrm{d}}{\mathrm{d}t}\left(\omega_{\hat{Q}}(\mathbf{q},t) \times \mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)\right)\frac{1 - \cos(\Omega_{\hat{Q}}(\mathbf{q})(s - s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\
&+ \sum_{P=0}^{\hat{Q}-1}\left[\frac{\mathrm{d}\mathbf{n}^{P\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}(\ell_P) + \frac{\mathrm{d}\mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}\frac{\sin(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right. \\
&\left.+ \frac{\mathrm{d}}{\mathrm{d}t}\left(\omega_P(\mathbf{q},t) \times \mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)\right)\frac{1 - \cos(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right]
\end{aligned} \tag{4.34}
$$

and $\dot{\mathbf{r}}^{SH}_{imp}$ as the "implicit" part,

$$
\dot{\mathbf{r}}^{SH}_{imp}(s,\mathbf{q}(t),t) = \sum_{P=0}^{\hat{Q}}\sum_{i=0}^{2}\mathbf{J}_{iP}(s,\mathbf{q},t)\dot{q}_{iP}. \tag{4.35}
$$

We can see how the initial values $(\mathbf{g}_i(t))_{i=0,1,2}$ and $\mathbf{h}(t)$ enter every time derivative through the first term of the sum in equation 4.34. The fact that they do so in an affine manner is not obvious, but is further discussed in [2].

Differentiating a second time in t, the implicit dependence on t through

$\mathbf{q}(t)$ complicates things even further:

$$\begin{aligned}
\ddot{\mathbf{r}}^{SH}(s,\mathbf{q},t) = {} & \frac{\mathrm{d}^2\mathbf{r}^0_{0,L}(t)}{\mathrm{d}t^2} + \frac{\mathrm{d}^2\mathbf{n}^{\hat{Q}\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t^2}(s - s^{\hat{Q}}_L) + \frac{\mathrm{d}^2\mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t^2}\frac{\sin(\Omega_{\hat{Q}}(\mathbf{q})(s - s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\
& + \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\omega_{\hat{Q}}(\mathbf{q},t) \times \mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)\right)\frac{1 - \cos(\Omega_{\hat{Q}}(\mathbf{q})(s - s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\
& + \sum_{P=0}^{\hat{Q}-1}\left[\frac{\mathrm{d}\mathbf{n}^{P\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}(\ell_P) + \frac{\mathrm{d}^2\mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t^2}\frac{\sin(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right. \\
& \left. + \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\omega_P(\mathbf{q},t) \times \mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)\right)\frac{1 - \cos(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right] \\
& + \sum_{P=0}^{\hat{Q}}\sum_{i=0}^{2}\mathbf{J}_{iP}(s,\mathbf{q},t)\frac{\mathrm{d}^2 q_{iP}}{\mathrm{d}t^2} \\
& + \sum_{P=0}^{\hat{Q}}\sum_{i=0}^{2}\frac{\mathrm{d}\mathbf{J}_{iP}(s,\mathbf{q},t)}{\mathrm{d}t}\dot{q}_{iP} \\
& + \sum_{P=0}^{\hat{Q}}\sum_{i=0}^{2}\sum_{\hat{P}=0}^{\hat{Q}}\sum_{j=0}^{2}\frac{\partial^2\mathbf{r}^{SH}(s,\mathbf{q},t)}{\partial q_{iP}\partial q_{j\hat{P}}}\dot{q}_{iP}\dot{q}_{j\hat{P}} \qquad (4.36)
\end{aligned}$$

Approximation In a bold maneuver, the two last second order terms in equation 4.36 were dropped, mainly due to time constraints and a wish to avoid code bloat. However, unyielding instabilities in the test implementations involving these explicit second order terms indicate a sensitivity to either accumulated roundoff error or some other numerical abberation. The remaining second order term includes $\ddot{q}_{iQ}$ and is part of the implicit solution of the system, and as such has an entirely different impact on the equations. Either way, the ubiquitous dependencies on curvature velocities suggest the dropped terms would at most contribute slightly to the motion of the Super-Helix and not affect the steady-state solution. Considering that the Euler-Lagrange equations essentially model a minimization problem [1], using an analogy to the reasoning in section 4.3 also argues for dropping the terms. As the matter is not finally settled, we would like to leave it open for future investigation.

Similar to the first time derivative case in equation 4.33 we end up with

an explicit part:

$$\begin{aligned}\ddot{\mathbf{r}}^{SH}_{exp}(s,\mathbf{q},t) = &\frac{\mathrm{d}^2\mathbf{r}^0_{0,L}(t)}{\mathrm{d}t^2} + \frac{\mathrm{d}^2\mathbf{n}^{\hat{Q}\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t^2}(s-s^{\hat{Q}}_L) + \frac{\mathrm{d}^2\mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t^2}\frac{\sin(\Omega_{\hat{Q}}(\mathbf{q})(s-s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\ &+ \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\omega_{\hat{Q}}(\mathbf{q},t)\times\mathbf{n}^{\hat{Q}\perp}_{0,L}(\mathbf{q},t)\right)\frac{1-\cos(\Omega_{\hat{Q}}(\mathbf{q})(s-s^{\hat{Q}}_L))}{\Omega_{\hat{Q}}(\mathbf{q})} \\ &+ \sum_{P=0}^{\hat{Q}-1}\left[\frac{\mathrm{d}\mathbf{n}^{P\parallel}_{0,L}(\mathbf{q},t)}{\mathrm{d}t}(\ell_P) + \frac{\mathrm{d}^2\mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)}{\mathrm{d}t^2}\frac{\sin(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right. \\ &\left.+ \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\omega_P(\mathbf{q},t)\times\mathbf{n}^{P\perp}_{0,L}(\mathbf{q},t)\right)\frac{1-\cos(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right] \quad (4.37)\end{aligned}$$

and an implicit one:

$$\ddot{\mathbf{r}}^{SH}_{imp}(s,\mathbf{q},t) = \sum_{P=0}^{\hat{Q}}\sum_{i=0}^{2}\mathbf{J}_{iP}(s,\mathbf{q},t)\ddot{q}_{iP} \quad (4.38)$$

4.2.3 Kinetic energy T, pt. 1

The kinetic energy is involved in two terms, $\frac{\mathrm{d}}{dt}\left(\frac{\partial T}{\partial\dot{q}_{iQ}}\right)$ and $\frac{\partial T}{\partial q_{iQ}}$. To start with the former, combining equations 4.33 and 4.26 and differentiating in time gives

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial\dot{q}_{iQ}}\right) &= \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial}{\partial\dot{q}_{iQ}}\left(\frac{1}{2}\int_0^L \rho S\left(\dot{\mathbf{r}}^{SH}(s,\mathbf{q},t)\right)^2\mathrm{d}s\right)\right] \\ &= \frac{\mathrm{d}}{\mathrm{d}t}\left[\rho S\int_0^L\left(\frac{\partial\dot{\mathbf{r}}^{SH}(s,\mathbf{q},t)}{\partial\dot{q}_{iQ}}\cdot\dot{\mathbf{r}}^{SH}(s,\mathbf{q},t)\right)\mathrm{d}s\right] \quad (4.39)\end{aligned}$$

As apparent from equation 4.33, differentiating $\dot{\mathbf{r}}^{SH}$ wrt. $\dot{q}_{iQ}$ only leaves the term $\mathbf{J}_{iQ}(s,\mathbf{q},t)$ from the double-sum, yielding:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_{iQ}}\right) &= \frac{\mathrm{d}}{\mathrm{d}t}\left[\rho S\int_0^L \left(\mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\dot{\mathbf{r}}^{SH}(s,\mathbf{q},t)\right)\mathrm{d}s\right] \\
&= \rho S\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\ddot{\mathbf{r}}_0\mathrm{d}s \\
&+ \rho S\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\left[\frac{\mathrm{d}^2\mathbf{n}_{0,L}^{Q\parallel}(\mathbf{q},t)}{\mathrm{d}t^2}(s-s_L^Q)\right. \\
&+ \frac{\mathrm{d}^2\mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)}{\mathrm{d}t^2}\frac{\sin(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})} \\
&+ \left.\frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\omega_Q(\mathbf{q},t)\times\mathbf{n}_{0,L}^{Q\perp}(\mathbf{q},t)\right)\frac{1-\cos(\Omega_Q(\mathbf{q})(s-s_L^Q))}{\Omega_Q(\mathbf{q})}\right]\mathrm{d}s \\
&+ \rho S\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\sum_{P=0}^{Q-1}\left[\frac{\mathrm{d}^2\mathbf{n}_{0,L}^{P\parallel}(\mathbf{q},t)}{\mathrm{d}t^2}(\ell_P)\right. \\
&+ \frac{\mathrm{d}^2\mathbf{n}_{0,L}^{P\perp}(\mathbf{q},t)}{\mathrm{d}t^2}\frac{\sin(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})} \\
&+ \left.\frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\omega_P(\mathbf{q},t)\times\mathbf{n}_{0,L}^{P\perp}(\mathbf{q},t)\right)\frac{1-\cos(\Omega_P(\mathbf{q})(\ell_P))}{\Omega_P(\mathbf{q})}\right]\mathrm{d}s \\
&+ \rho S\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\sum_{j\hat{Q}}\mathbf{J}_{j\hat{Q}}(s,\mathbf{q},t)\ddot{q}_{j\hat{Q}}\mathrm{d}s \qquad (4.40)
\end{aligned}$$

As discussed in section 4.2.2, the shape of the gradient $\mathbf{J}_{iQ}(s,\mathbf{q},t)$ depends on the piecewise constant curvatures. Since each sum in equation 4.40 contains at least one gradient, the entire expression assumes the same kind of piecewise appearance, making it very convenient to split the integrals into N pieces (N as usual being the number of segments). In this way, each term in the sums has its own initial values, including derivatives with respect to previous segments. With the exception of the last term, all terms in 4.40 are of the form $\int \mathbf{J}\cdot\mathbf{a}[s,\mathbf{q},t]\ \mathrm{d}s$ where $\mathbf{a}$ is some externally imposed vector valued function given explicitly in terms of some or all of s, $\mathbf{q}$ and t. These terms are appropriate for straightforward integration, either explicitly or -

as in our case - symbolically. For example, the first term can be written

$$\rho S \int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot \ddot{\mathbf{r}}_0 \mathrm{d}s = \rho S \sum_{\hat{Q}=0}^{Q-1} \int_{s_L^{\hat{Q}}}^{s_R^{\hat{Q}}} \mathbf{0}\ \mathrm{d}s\cdot \ddot{\mathbf{r}}_0$$
$$+\rho S \int_{s_L^Q}^{s_R^Q} \mathbf{J}_{iQ}(s,\mathbf{q},t)\ \mathrm{d}s\cdot \ddot{\mathbf{r}}_0 + \rho S \sum_{\hat{Q}=Q+1}^{N} \left[\int_{s_L^{\hat{Q}}}^{s_R^{\hat{Q}}} \tilde{\mathbf{J}}_{iQ}(s,\mathbf{q},t)\ \mathrm{d}s\cdot \ddot{\mathbf{r}}_0 \right] \tag{4.41}$$

which holds three terms, each corresponding to one of the three cases for $\mathbf{J}_{iQ}(s,\mathbf{q},t)$ described in section 4.2.2. All of these single gradient terms are stuck into the RHS "kitchen sink" vector $\mathbf{A}$. For more involved $\mathbf{a}$ that cannot be brought outside of the integral easily (e.g. those that contain derivatives of initial values of the segments), the dot product is performed symbolically before integration, leading to an extensive set of integral routines for all possible cases. The necessary time derivatives of $\mathbf{n}$, $\mathbf{r}$ etc. are computed recursively using equations 4.18 and 4.20, as further described in chapter 5.

The last term in 4.40 is very important, as it is the only term in the equations that contains $\ddot{\mathbf{q}}$. Again observing the cases for $\mathbf{J}_{iQ}(s,\mathbf{q},t)$ from section 4.2.2, it can be expanded to

$$\begin{aligned}
&\int_0^L \mathbf{J}_{iQ} \sum_{i'\hat{Q}} \mathbf{J}_{i'\hat{Q}} \ddot{q}_{i'\hat{Q}} \mathrm{d}s = \\
&\quad \rho S \sum_{\hat{Q}=0}^{Q-1} \int_{s_L^{\hat{Q}}}^{s_R^{\hat{Q}}} \mathbf{0}\cdot \sum_{P=0}^{\hat{Q}} \sum_{i'}^{2} \tilde{\mathbf{J}}_{i'P}(s,\mathbf{q},t)\ddot{q}_{i'P}\mathrm{d}s \\
&\quad + \rho S \int_{s_L^Q}^{s_R^Q} \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot \sum_{P=0}^{Q-1} \sum_{i'=0}^{2} \tilde{\mathbf{J}}_{i'P}(s,\mathbf{q},t)\ddot{q}_{i'P}\mathrm{d}s \\
&\quad + \rho S \int_{s_L^Q}^{s_R^Q} \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot \sum_{i'=0}^{2} \mathbf{J}_{i'Q}(s,\mathbf{q},t)\ddot{q}_{i'Q}\mathrm{d}s \\
&\quad + \rho S \sum_{\hat{Q}=Q+1}^{N} \int_{s_L^{\hat{Q}}}^{s_R^{\hat{Q}}} \tilde{\mathbf{J}}_{iQ}(s,\mathbf{q},t)\cdot \sum_{P=0,P\neq Q}^{\hat{Q}-1} \sum_{i'=0}^{2} \tilde{\mathbf{J}}_{i'P}(s,\mathbf{q},t)\ddot{q}_{i'P}\mathrm{d}s \\
&\quad + \rho S \sum_{\hat{Q}=Q+1}^{N} \int_{s_L^{\hat{Q}}}^{s_R^{\hat{Q}}} \tilde{\mathbf{J}}_{iQ}(s,\mathbf{q},t)\cdot \sum_{i'=0}^{2} \mathbf{J}_{i'\hat{Q}}(s,\mathbf{q},t)\ddot{q}_{i'\hat{Q}}\mathrm{d}s \\
&\quad + \rho S \sum_{\hat{Q}=Q+1}^{N} \int_{s_L^{\hat{Q}}}^{s_R^{\hat{Q}}} \tilde{\mathbf{J}}_{iQ}(s,\mathbf{q},t)\cdot \sum_{i'=0}^{2} \tilde{\mathbf{J}}_{i'Q}(s,\mathbf{q},t)\ddot{q}_{i'Q}\mathrm{d}s
\end{aligned} \tag{4.42}$$

Here the terms get more numerous, due to the involvement of *two* gradients. We take the summation signs outside of the integrals to indicate that each integration is performed separately, depending on the combination of gradients. Examining the expression closer, one in fact realizes that it would be very suitable to write it in matrix form. For example, the different combinations of gradients wrt. curvatures from the same and from different segments map nicely to *diagonal* terms and *cross* terms, respectively. Indeed, this is a central part of the Super-Helix derivations, and it is how the symmetric, $3N \times 3N$ *mass matrix* $\mathbb{M}$ of the system is composed. By writing the whole expression as a dot product, we obtain

$$\int_0^L \mathbf{J}_{iQ} \sum_{\hat{Q}=0}^{N} \sum_{j=0}^{2} \mathbf{J}_{j\hat{Q}} \ddot{q}_{j\hat{Q}} \mathrm{d}s = \mathbf{m}_{iQ} \cdot \ddot{\mathbf{q}}(t) \tag{4.43}$$

where $\mathbf{m}_{iQ}$ is a $3N$ row vector, each element $i'\hat{Q} = [0..3N]$ containing the integral $\int \mathbf{J}_{iQ}\mathbf{J}_{i'\hat{Q}}\ \mathrm{d}s$, still computed in a piecewise fashion as described above. Stacking $3N$ of these dot products, corresponding to all segments and curvatures of the Super-Helix, we have the $3N$ row vectors $\mathbf{m}_{iQ}$ of the mass matrix. Thus element $(iQ, i'\hat{Q})$ describes some kind of relation between the i:th curvature of the Q:th element and the i':th curvature of the $\hat{Q}$:th element. It should be fairly straightforward to see how this implies that the mass matrix is symmetric. All in all, we have factored out $\ddot{\mathbf{q}}$ from the kinetic energy, taking a big step towards enabling us to discretize the system in time and solve for the change in curvatures. More on that in section 4.2.8.

4.2.4 Kinetic energy T, pt. 2

For the second term of 4.25 involving kinetic energy, $\frac{\partial T}{\partial q_{iQ}}$, the derivation is similiar to that in equation 4.40:

$$\frac{\partial T}{\partial q_{iQ}} = \rho S \int_0^L \left(\frac{\partial \dot{\mathbf{r}}^{SH}(s, \mathbf{q}, t)}{\partial q_{iQ}} \cdot \dot{\mathbf{r}}^{SH}(s, \mathbf{q}, t) \right) \mathrm{d}s \tag{4.44}$$

Alas, differentiating $\dot{\mathbf{r}}^{SH}(s, \mathbf{q}, t)$ wrt. $\mathbf{q}$ does not work out quite as nicely as $\dot{\mathbf{q}}$ did. Using equations 4.30 and 4.33 and the Chain Rule,

$$\frac{\partial T}{\partial q_{iQ}} = \rho S \frac{1}{2} \int_0^L \frac{\partial}{\partial q_{iQ}} \left(\sum_{\tilde{Q}=0}^{N} \sum_{k=0}^{2} \mathbf{J}_{k\tilde{Q}}(s, \mathbf{q}, t) \dot{q}_{k\tilde{Q}} + \dot{\mathbf{r}}^{SH}_{exp}(s, \mathbf{q}, t) \right) \cdot \left(\sum_{\hat{Q}=0}^{N} \sum_{j=0}^{2} \mathbf{J}_{j\hat{Q}}(s, \mathbf{q}, t) \dot{q}_{j\hat{Q}} + \dot{\mathbf{r}}^{SH}_{exp}(s, \mathbf{q}, t) \right) \mathrm{d}s \tag{4.45}$$

Since $\frac{\partial \dot{q}_{jQ'}}{\partial q_{iQ}} = 0 \quad \forall i, j, Q, Q'$, what remains is

$$\frac{\partial T}{\partial q_{iQ}} = \rho S \frac{1}{2} \int_0^L \sum_{\tilde{Q}=0}^{N} \sum_{k=0}^{2} \sum_{\hat{Q}=0}^{N} \sum_{j=0}^{2} \frac{\partial \mathbf{J}_{k\tilde{Q}}(s,\mathbf{q},t)}{\partial q_{iQ}} \dot{q}_{k\tilde{Q}} \cdot \left(\mathbf{J}_{j\hat{Q}}(s,\mathbf{q},t)\dot{q}_{j\hat{Q}} + \dot{\mathbf{r}}^{SH}_{exp}(s,\mathbf{q},t) \right) \mathrm{d}s \tag{4.46}$$

As described in section 4.2.2, discarding second order terms is still a slightly controversial issue. We did however proceed in this case as well, causing the entire equation 4.46 to vanish from the equations. See the paragraph following equation 4.40 for the reasoning behind this decision.

4.2.5 Potential energy U

For the potential energy U we use a linear spring force in curvature space, with equilibrium in the natural curvatures κ_i^N and spring constants $(EI)_{iQ}$ derived by Bertails *et al.* [3] from standard mechanical engineering formulae for stiffness in bars:

$$\begin{aligned}
\frac{\partial U}{\partial q_{iQ}} &= \frac{\partial}{\partial q_{iQ}} \left(\frac{1}{2} \int_0^L \sum_{j=0}^{2} (EI)_j \left(\kappa_j^{SH}(s,\mathbf{q}) - \kappa_j^n(s) \right)^2 \right) \mathrm{d}s \\
&= \int_0^L \sum_{j=0}^{2} (EI)_j \underbrace{\left(\kappa_j^{SH}(s,\mathbf{q}) - \kappa_j^n(s) \right)}_{=0 \text{ for } s \notin S_Q} \underbrace{\left(\frac{\partial}{\partial q_{iQ}} \kappa_j^{SH}(s,\mathbf{q}) \right)}_{=0 \text{ for } i \neq j} \mathrm{d}s \\
&= \int_{s_L^Q}^{s_R^Q} (EI)_i \left(\kappa_i^{SH}(s,\mathbf{q}) - \kappa_i^n(s) \right) \mathrm{d}s
\end{aligned} \tag{4.47}$$

Here we assume κ_i^n is constant in s over the segment, just like κ_i^{SH}. For any segment S_Q, $\int (EI)_i \, \mathrm{d}s = \ell_Q (EI)_i$ can be factored out into the 3×3 diagonal *stiffness matrix* $\mathbb{K}_Q$, and the κ:s be put in the 3-vectors $\mathbf{q}_Q$ and $\mathbf{q}_Q^n$, giving us:

$$\frac{\partial U}{\partial \mathbf{q}_Q} = \mathbb{K}_Q (\mathbf{q}_Q(t) - \mathbf{q}_Q^n) \tag{4.48}$$

For the entire system, these vector expressions stack to form

$$\frac{\partial U}{\partial \mathbf{q}} = \mathbb{K} (\mathbf{q}(t) - \mathbf{q}^n) \tag{4.49}$$

with $\mathbb{K}$ still diagonal.

4.2.6 Dissipation potential D

As stated in section 4.2.1, there are two different possible models of the dissipation potential (4.28 and 4.29). Both essentially act as dampers, i.e. a force in the opposite direction of the velocity (in curvature space), the difference being that in 4.28 the term goes into the RHS **A** vector, like:

$$
\begin{aligned}
D(\mathbf{q},\dot{\mathbf{q}},t) &= \frac{1}{2}\int_0^L \gamma \sum_{i=0}^{2} \left(\dot{\kappa}_i^{SH}(s,\mathbf{q})\right)^2 \mathrm{d}s \\
\frac{\partial D(\mathbf{q},\dot{\mathbf{q}},t)}{\partial \dot{q}_{iQ}} &= \gamma \int_0^L \sum_{j=0}^{2} \dot{\kappa}_j^{SH}(s,\mathbf{q}) \underbrace{\frac{\partial}{\partial \dot{q}_{iQ}} \dot{\kappa}_j^{SH}(s,\mathbf{q})}_{=0 \text{ for } i\neq j \text{ or } s\notin S_Q} \mathrm{d}s \\
&= \gamma \int_{s_L^Q}^{s_R^Q} \dot{\kappa}_i^{SH}(s,\mathbf{q})\, \mathrm{d}s
\end{aligned}
$$

Now the result for segment S_Q can be written as a 3-vector:

$$
\frac{\partial D}{\partial \dot{\mathbf{q}}_Q} = \gamma \ell_Q \dot{\mathbf{q}}_Q(t)
$$

which, again, for the entire system amounts to

$$
\frac{\partial D}{\partial \dot{\mathbf{q}}} = \gamma \mathbb{L} \dot{\mathbf{q}}(t) \tag{4.50}
$$

where $\mathbb{L}$ is a $3N \times 3N$ diagonal matrix with triplets of the N segment lengths ℓ_Q on the diagonal. The term from 4.29 on the other hand actually ends up on the LHS, which has a considerable effect on the system (see section 4.2.9). Using the same pattern as in section 4.2.5,

$$
\begin{aligned}
D(\mathbf{q},\dot{\mathbf{q}},t) &= \frac{1}{2}\int_0^L \mu \sum_{i=0}^{2} (EI)_i \left(\dot{\kappa}_i^{SH}(s,\mathbf{q})\right)^2 \mathrm{d}s \\
\frac{\partial D(\mathbf{q},\dot{\mathbf{q}},t)}{\partial \dot{q}_{iQ}} &= \cdots = \mu \int_{s_L^Q}^{s_R^Q} (EI)_i \dot{\kappa}_i^{SH}(s,\mathbf{q})\, \mathrm{d}s \\
\implies \frac{\partial D}{\partial \dot{\mathbf{q}}} &= \mu \mathbb{K} \dot{\mathbf{q}}
\end{aligned} \tag{4.51}
$$

4.2.7 External forces F

The external forces $\mathbf{F}(s,t)$ acting on the Super-Helix are written

$$
\mathbf{F}(s,t) = \rho S \mathbf{g} - \nu \dot{\mathbf{r}}^{SH}(s,\mathbf{q},t) + \mathbf{F}^i(s,t) \tag{4.52}
$$

where $\mathbf{g}$ is gravity, ν is an air resistance coefficient and $\mathbf{F}^i(s,t)$ are the interaction forces imposed by collisions and other external interaction with the Super-Helix. In order to feed these Cartesian-space forces into the system, each of them needs to be transformed into generalized forces through the integral $\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\mathbf{f}\ \mathrm{d}s$. This integral is one of the keys to the globalness of the Super-Helix, as each force contribution is computed against *all* curvatures in the model, not just the local ones.

When computing $\int \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\mathbf{F}(s,t)\ \mathrm{d}s$, the gravity force $\rho S\mathbf{g}$ is constant in space and time and goes outside of the integral. The air drag $\nu\dot{\mathbf{r}}^{SH}(s,\mathbf{q})$ is slightly more cumbersome. Using equations 4.33 and 4.34,

$$\begin{aligned}
&\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\nu\dot{\mathbf{r}}^{SH}(s,\mathbf{q},t)\ \mathrm{d}s = \nu\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot(\dot{\mathbf{r}}^{SH}_{exp}(s,\mathbf{q},t)+\dot{\mathbf{r}}_{imp}(s,\mathbf{q}))\ \mathrm{d}s\\
&=\nu\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\dot{\mathbf{r}}^{SH}_{exp}(s,\mathbf{q},t)\ \mathrm{d}s+\nu\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\sum_{\hat{Q}=0}^{N}\sum_{j=0}^{2}\mathbf{J}_{j\hat{Q}}(s,\mathbf{q},t)\dot{q}_{j\hat{Q}}\ \mathrm{d}s\\
&=\nu\int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\dot{\mathbf{r}}^{SH}_{exp}(s,\mathbf{q},t)\ \mathrm{d}s+\mathbf{m}_{iQ}\cdot\dot{\mathbf{q}}(t) \qquad (4.53)
\end{aligned}$$

where $\mathbf{m}_{iQ}$ again is the $3N$ row vector of the mass matrix $\mathbb{M}$, corresponding to q_{iQ}. Using equation 4.34 and the approach to piecewise integration presented in section 4.2.2, the first term is also computed symbolically.

Finally, the interaction forces $\mathbf{F}^i(s,t)$ need to be integrated numerically. This was left as an input to the system, depending on which collision system will be employed in the pipeline. However, an example implementation is briefly presented in section 4.2.10.

4.2.8 Temporal discretization

In order to advance the system in time one not only needs to perform a spatial discretization, but also a temporal one. From previous sections, we have

$$\frac{\mathrm{d}}{dt}\left(\frac{\partial T}{\partial\dot{q}_{iQ}}\right)-\frac{\partial T}{\partial q_{iQ}}+\frac{\partial U}{\partial q_{iQ}}+\frac{\partial D}{\partial\dot{q}_{iQ}}=\int_0^L \mathbf{J}_{iQ}[s,\mathbf{q},t]\cdot\mathbf{F}[s,t]\mathrm{d}s \qquad (4.54)$$

which for the entire system now can be replaced by

$$\rho S\mathbb{M}[s,\mathbf{q}]\ddot{\mathbf{q}}+\mathbb{K}(\mathbf{q}-\mathbf{q}^n)+\mu\mathbb{K}\dot{\mathbf{q}}=\mathbf{A}[s,\mathbf{q},\dot{\mathbf{q}}]+\mathbf{Q}^{int}[s,\mathbf{q},t] \qquad (4.55)$$

which, thanks to the symbolic integration scheme presented previously in this chapter, in fact is a spatially discretized system. The first term holds the mass matrix and $\ddot{\mathbf{q}}$ as derived from the first kinetic term of equation 4.54, while the second term holds the stiffness matrix and curvature displacement as derived from the potential energy. $\mu\mathbb{K}\ddot{\mathbf{q}}$ was derived from the

diffusion term. The the right hand side vector $\mathbf{A}$ holds all explicitly given, but symbolically integrated, terms emanating from gravity and manipulation of the root of the Super-Helix. It includes some amount of intertia from the kinetic term and also air drag, and is given by

$$\mathbf{A}_{iQ}[s,\mathbf{q},\dot{\mathbf{q}}] = \int_0^L \mathbf{J}_{iQ}[s,\mathbf{q},t]\cdot\left(-\rho S\ddot{\mathbf{r}}^{SH}_{exp}[s,\mathbf{q},t] + \nu\dot{\mathbf{r}}^{SH}_{exp}[s,\mathbf{q},t] + \rho S\mathbf{g}\right)\,\mathrm{d}s + \nu\mathbb{M}\dot{\mathbf{q}} \tag{4.56}$$

Here $\ddot{\mathbf{r}}^{SH}_{exp}$ and $\dot{\mathbf{r}}^{SH}_{exp}$ are the "explicit" parts of the time derivatives of $\mathbf{r}^{SH}$, as given by 4.34 and 4.37. The "implicit" parts constitute $\mathbb{M}$. Finally, $\mathbf{Q}^{int}$ is the only numerically integrated term, dealing with external forces mainly from collision response:

$$\mathbf{Q}^{int}_{iQ}[s,\mathbf{q},t] = \int_0^L \mathbf{J}_{iQ}[s,\mathbf{q},t]\cdot\mathbf{F}_i\,\mathrm{d}s. \tag{4.57}$$

Like Bertails *et al.* we choose to apply a semi-implicit Newton scheme to this system. The term "semi-implicit" refers to the fact that some of the occurrences of $\mathbf{q}$ are treated explicitly, while some are involved in the actual implicit timestep. In this section, we have followed Bertails *et al.*'s example and denote explicitly computated terms with brackets (i.e. [ ]) while all other $\mathbf{q}$:s are treated implicitly. In order to apply the scheme, we perform the change of variables $\mathbf{x} = \mathbf{q}$ and $\mathbf{y} = \dot{\mathbf{q}}$. Inserting this into equation 4.55 yields

$$\dot{\mathbf{x}} = \mathbf{y} \tag{4.58}$$

$$\rho S\mathbb{M}\dot{\mathbf{y}} + \mathbb{K}(\mathbf{x}-\mathbf{x}^n) + \mu\mathbb{K}\mathbf{y} = \mathbf{A} + \mathbf{Q}^{int}. \tag{4.59}$$

Rewriting $\dot{\mathbf{x}}$ and $\dot{\mathbf{y}}$ above in terms of first-order backward differences, and rearranging, we get

$$\frac{\mathbf{x}_{t+1}-\mathbf{x}_t}{\varepsilon} - \mathbf{y}_{t+1} = 0 \tag{4.60}$$

$$\mathbb{M}\frac{\mathbf{y}_{t+1}-\mathbf{y}_t}{\varepsilon} + \mu\mathbb{K}\mathbf{y}_{t+1} + \mathbb{K}\mathbf{x}_{t+1} = (\mathbf{A}+\mathbf{Q})_t + \mathbb{K}\mathbf{x}^n \tag{4.61}$$

Coupling these equations we obtain the $6N \times 6N$ system

$$\begin{pmatrix} \mathbf{I} & -\varepsilon\mathbf{I} \\ \varepsilon\mathbb{K} & \mathbb{M}+\varepsilon\mu K \end{pmatrix}\begin{pmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_t \\ \mathbb{M}\mathbf{y}_t + \varepsilon([\mathbf{A}+\mathbf{Q}]_t + \mathbb{K}\mathbf{x}^n) \end{pmatrix}$$

which can be reduced to a $3N \times 3N$ system using the substitutions [2]

$$\begin{aligned} \mathbf{y}_{t+1} &= \frac{\mathbf{x}_{t+1}-\mathbf{x}_t}{\varepsilon} \\ \Delta\mathbf{x} &= \mathbf{x}_{t+1}-\mathbf{x}_t \\ \Delta\mathbf{y} &= \mathbf{y}_{t+1}-\mathbf{y}_t \\ \Delta\mathbf{y} &= -\mathbf{y}_t + \frac{\Delta x}{\varepsilon} \end{aligned}$$

The final system becomes

$$\big(\mathbb{M} + \varepsilon\mu\mathbb{K} + \varepsilon^2\mathbb{K}\big)\Delta\mathbf{x} = \varepsilon\Bigg(\mathbb{M}\mathbf{y}_t - \varepsilon\big(\big[\mathbf{A} + \mathbf{Q}^{int}\big]_t + \mathbb{K}(\mathbf{q}^n - \mathbf{x}_t)\big)\Bigg) \tag{4.62}$$

4.2.9 Robustness and stability

Since all spatial integrals except for the ones related to collisions are performed symbolically, any numerical error pertaining to the spatial integration will assumedly be in the order of the round-off error. This is one of the big pay-offs from the tedious math derivations presented in this thesis.

Having been temporally discretized using a semi-implicit Newton scheme, the system 4.62 is reasonably although not unconditionally stable in time. [3] claims that $1/30s$ is a decent time step size, although we went even further without seeing any time-related instabilities, save for in the experimental collision detection. Even at $1/4s$ no oscillations or transients were evident, as long as the LHS matrix remained well conditioned. However, being semi-implicitly integrated, the detail of the motion is closely related to the temporal resolution. In a sense, taking too large time steps corresponds to low-pass filtering the system, removing interesting high-frequency modes.

On the note of conditioning, the system does display a set of properties that are critical to its stability and well worth examining. First and foremost, the mass matrix is not positive definite on its own. In fact, it has rank $2N$ in a $3N$ system. This is in broad terms caused by the assumption of an infinitesimal cross-section area of the hair, causing a lack of torsional inertia and no stress due to torque around the tangent. This means that torsional modes propagate through the Super-Helix at infinite speeds, and that the curvatures $(\kappa_i)_{i=0,1,2}$ are not strictly independent [2]. Also, compared to the Frenet-Serret equations discussed in section 4.1.1, which can describe any curve locally using one torsion and *one* curvature, the Cosserat model has *two* curvatures. This causes some amount of redundancy in the formulation, but given the intervals of constant curvature and enforced initial material frames it is crucial to provide sufficient flexibility in the model (see Figure 4.2).

Luckily, if the proper damping term is chosen (see 4.2.6), the implicit time integration scheme (4.2.8) results in the left hand side $(\mathbb{M} + (\varepsilon\mu + \varepsilon^2)\mathbb{K})$ where the second term acts like a damper. The more dominant it is the more similar to a diagonal matrix does the $\mathbb{A}$ become, essentially decoupling the system and diminishing the energy contributed by the RHS forces. In the limit the whole system reduces to $\Delta\mathbf{x} = \mathbf{0}$. Note that this is only the case when using 4.28 for the dissipation potential term. When instead using 4.29, the final term ends up on the RHS and does not contribute to the

damping of the system $\mathbb{A}$, only of the particular solution $\mathbf{x} = \mathbb{A}\mathbf{b}$.

Unfortunately, the balancing point between mass matrix and stiffness matrix, i.e. when the system gets negative eigenvalues and becomes indefinite, occurs slightly sooner than wanted. The cause for this is unknown, and not even extensive efforts to alleviate the problem gave any significant results. The symptom is an undesirable stiffness in the hair strands and an exhaggerated tendency to return to the natural state. The only remedy, to instead increase μ and thus the damping, leads to an attenuation of high-frequency modes and an overly damped solution.

4.2.10 Collision handling

For the sake of verifying the implementation, a simple collision handling algorithm was implemented. It is entirely based on penalty forces, and quickly revealed the great difficulties related to handling collisions robustly in generalized coordinates. The quadratic reduction technique used by Bertails *et al.* in [2] provided at least some measure of stability, although it is fairly sensitive to the parameters involved. Once a penetration of depth δ is detected, the normalized direction $\mathbf{n}_c$ to the closest surface point is computed, and the force in that direction is given by

$$\begin{aligned} \text{if} \quad \delta \leq 0 \quad & \mathbf{F}_R = \mathbf{0} \\ \text{if} \quad 0 \leq \delta \leq \delta_{reg} \quad & \mathbf{F}_R = \frac{k_c \delta^2}{2\delta_{reg}} \mathbf{n}_c \\ \text{else} \quad & \mathbf{F}_R = k_c(\delta - \frac{\delta_{reg}}{2})\mathbf{n}_c \end{aligned}$$

where δ_{reg} is the "regulation depth" and k_c is a spring stiffness constant. For a sphere of radius 1 a good value for δ_{reg} was empirically determined to around 0.1, while k_c depends on the weight-stiffness scaling of the hair strand.

A spherical proxy object of variable size was implemented. For ease of implementation, speed and improved stability (but at the obvious cost of realism), only the endpoints of each strand were checked for collisions. The results were encouraging, mainly in terms of the absence of bouncing. However a fairly tight timestep restriction applies; for any kind of stability the empirically determined timestep was smaller than $1/128s$ to avoid spurious force peaks due to exhaggerated penetration.

4.3 Modeling

Being formulated in a parameter space vastly different from Euclidean space, Super-Helices are highly unintuitive to interact with directly, for example

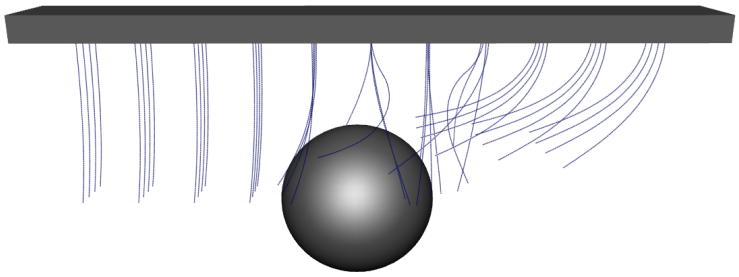

Figure 4.5: The proxy sphere has entered at considerable speed from the right hand side, stopping halfway through the brush. Strands reach a steady state without significant bouncing etc. Timestep size $1/128s$.

in a modeling environment. Straightfoward manipulation of the curvature values maps poorly to perceived change in shape in $\mathbb{R}^3$. For example, there exist no "up" or "left" directions, only κ_0, κ_1 and κ_2. Additionally each change in curvature of a certain helical segment affects not only the shape of that particular segment, but also the initial condition for all subsequent segments, effectively imposing an affine transform on them. Hence there exists no nice way to translate only a local neighborhood around a point, but any change has a global effect.

Whereas the latter problem is not easily addressed - hard constraints are inherently hard to impose on a system in generalized coordinates [26] - the former one can be mitigated using standard non-linear minimization techniques, cf. [20]. Consider a segment S_Q. Translating a point on it through $\mathbb{R}^3$ can only be achieved by altering the curvatures of either S_Q, some preceeding segment $S_{Q'}, Q' < Q$ or a combination of those. As should be clear, the curvatures of subsequent segments $S_{Q''}, Q'' > Q$ have no effect whatsoever on S_Q.

In general, the functional describing the error between the desired location $\mathbf{Y}$ in Euclidean space of some point s on the Super-Helix and its current location $\mathbf{r}(s, \mathbf{q})$ can be written

$$\chi^2(s, \mathbf{q}) = (\mathbf{Y} - \mathbf{r}(s, \mathbf{q}))^2. \tag{4.63}$$

Differentiating this functional with respect to the curvature $q_{iQ}, i = 0, 1, 2$ of any segment S_Q with its initial point $s' < s$ in the Super-Helix (see 4.30), we obtain the Gradient $J_{iQ}(s, \mathbf{q})$ and Hessian $H_{iQ,jQ}(s, \mathbf{q})$. Writing the

measured error $\mathbf{Y} - \mathbf{r}(s, \mathbf{q})$ as $\epsilon(s, \mathbf{q})$,

$$J_{iQ} = \frac{\partial \chi^2(s, \mathbf{q})}{\partial q_{iQ}} = -2\epsilon(s, \mathbf{q}) \frac{\partial \mathbf{r}(s, \mathbf{q})}{\partial q_{iQ}} \tag{4.64}$$

$$H_{iQ,jQ} = \frac{\partial^2 \chi^2(s, \mathbf{q})}{\partial q_{iQ} \partial q_{jQ}} = 2 \frac{\partial \mathbf{r}(s, \mathbf{q})}{\partial q_{iQ}} \frac{\partial \mathbf{r}(s, \mathbf{q})}{\partial q_{jQ}} - 2\epsilon(s, \mathbf{q}) \frac{\partial^2 J}{\partial q_{iQ} \partial q_{jQ}}. \tag{4.65}$$

As [20] explains, the second order term in $H_{iQ,jQ}$ does at best contain uncorrelated measurement noise - which in our case should be negligible - and at worst destabilizing terms due to a poorly fitting model. Thus we, too, follow the convention and ignore this term.

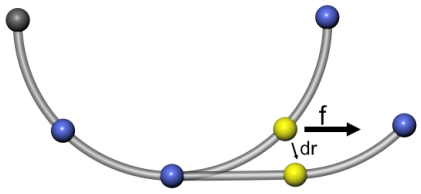

Figure 4.6: The yellow segment endpoint affected by a translational "force" $\mathbf{f}$, subject to constraint in arclength and with freedom in curvatures of the preceeding segment. The algorithm will find the optimal translation $\delta\mathbf{r}$ to accomodate the requested translation in a for the user intuitive manner. Blue spheres are free endpoints, grey sphere is the clamped initial point.

Now, assuming $\mathbf{Y}$ and $\mathbf{r}$ are sufficiently close (as is the case in our application, see section 5.3.1), the functional χ^2 is well approximated by the quadratic form

$$\chi^2(\mathbf{q}) \approx \gamma - \mathbf{J} \cdot \mathbf{q} + \frac{1}{2} \mathbf{q} \cdot \mathbb{H} \cdot \mathbf{q}. \tag{4.66}$$

Thus we can use the well known ([20])

$$\mathbf{q}_{\min} = \mathbf{q}_{\text{cur}} + \mathbb{H}^{-1} \left[-\nabla \chi^2(\mathbf{q}_{\text{cur}}) \right] \tag{4.67}$$

to take a single leap in curvature space to a configuration that minimizes ϵ, effectively bending the specified segment(s) to accommodate the requested translation (see figure 4.6).
If ϵ is large enough for the parameter space to become poorly approximated by 4.66, the application of 4.67 may need to be preceeded by steepest descent steps. Resorting to the Levenberg-Marquardt method is one option for a seamless blend between the two [20].

Chapter 5

Implementation

Given the results in chapter 4, we now show how the actual implementation was performed. All Super-Helix mathematics was implemented as a standalone library, and a user interface was provided using the Maya API.

5.1 Math

5.1.1 Symbolic integration

A major undertaking in this thesis was to implement the symbolic integration as briefly sketched in [3] and given some further - but still insufficient - attention in [2]. As shown in 4.2, at the core of every explicit term in the discretized Euler-Lagrange equations is a spatial integral in s over some combination of the gradients $\mathbf{J}_{iQ}(s, \mathbf{q}, t), i = 0, 1, 2, Q = 1..N$.

Examining the structure of the system, it was clear that many of the variables defined in sections 4.1.2 and 4.2.2 would be subject to extensive reuse. For example, the normalized Darboux vector ω is present in nearly every term in every integral. To enable some amount of code reuse we designed a class hierarchy where each specific integral would inherit its necessary values from base cases of simpler integrals. For example $\int \mathbf{J}_{iQ}(s, \mathbf{q}, t)\mathbf{J}_{jP}(s, \mathbf{q}, t)\ \mathrm{d}s$, where Q is the current segment and P is a previous segment, would inherit all of its members from the base cases $\mathbf{J}_{iQ}(s, \mathbf{q}, t)$ and $\mathbf{J}_{iP}(s, \mathbf{q}, t)$. Virtual inheritance and explicit base class constructor calls were in many cases necessary to avoid the "diamond problem", i.e. ambiguous overloads (see Figure 5.1). This problem would for example surface when $\mathbf{J}_{iQ}(s, \mathbf{q}, t)$ and $\mathbf{J}_{iP}(s, \mathbf{q}, t)$ from the previous example both inherit the base class, containing declaration and initialization of ω, Ω and other generic variables. Inheriting both of these gradient classes into the class providing the $\int \mathbf{J}_{iQ}(s, \mathbf{q}, t)\mathbf{J}_{jP}(s, \mathbf{q}, t)\ \mathrm{d}s$ integral required a virtual inheritance from the base class, and an explicit call to the base class constructor from the integral constructor.

Each integral requires certain initial values. In the case of $\int \mathbf{J}_{iQ}(s, \mathbf{q}, t) \cdot \mathbf{F}\ \mathrm{d}s$,

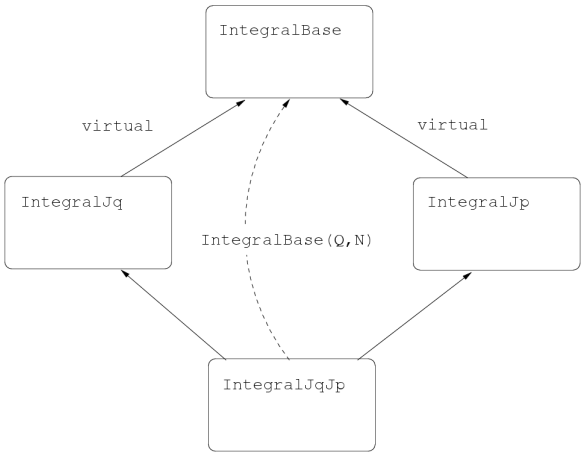

Figure 5.1: A typical "diamond" situation remedied by using virtual inheritance and an explicit base class initialization call. The `Jp` and `Jq` abbreviations refer to gradients with respect to curvatures of preceeding and current segments, respectively.

integrating over segment S_Q only requires the initial coordinate frame $\mathbf{n}_{i,L}^{Q}$ and the local curvatures $\mathbf{q}^Q$, whereas integrating the same gradient $\mathbf{J}_{iQ}(s, \mathbf{q}, t)$ over subsequent segments $S'_Q, Q' = Q + 1..N$ also requires the derivatives of the initial frame of reference with respect to the curvatures of S_Q. This is a fairly intuitive result of the sequential nature of the Super-Helix, as discussed in section 4.1.2. It should be obvious that any possible integral over derivatives of the i:th segment with respect to any combination of previous curvatures requires initial values from at most $i - 1$ segments. As a preprocessing step in each frame we use 4.30 and 4.32 to compute these initial values and store them in the matrices $(\mathbb{D}_i)_{i=0,1,2}$. For example, in the matrix $\mathbb{D}_0$ containing derivatives of the initial tangent, element (Q, Q') holds the derivatives of the Q:th initial tangent wrt. the Q':th segment's curvatures $q_{jQ'}, j = 0, 1, 2$. Actually, the combined gradients for $\forall j$ form a 3×3 Jacobian matrix, since they describe the derivative of a vector wrt. another vector. For optimization purposes, it follows from previous reasoning that all elements where $Q < Q'$ are indentical to 0, so the matrices $\mathbb{D}_i$ are all triangular.

With the initial value derivatives established, building the mass matrix $\mathbb{M}$ and the generalized force vector $\mathbf{A}$ boils down to performing the segment-wise integrals presented in e.g. equation 4.42. We initially used numerical methods for differentiation and integration, but once the results were verified we employed the symbolic powers of MATHEMATICA, in order to increase

stability, accuracy and efficiency.

Unfortunately, most - if not all - of the results from straightforward integration in Mathematica involve divisions by different powers of Ω, a fact that calls for special treatment of the degenerate case $\mathbf{q}_Q = \mathbf{0}$ (i.e. a straight, non-twisting curve). The typical way to deal with this would be to perform some kind of interpolation over a vicinity around the origin, but that had to be ommitted due to time constraints. Instead we observed that by going from single to double precision in the computations, the singularity breakpoint could be pushed sufficiently close to 0 for the singularities to rarely bear significance in practical circumstances. It however calls for further treatment before the implementation sees use in production.

5.1.2 Linear solvers

With the complete system as described in section 4.2.8 at hand, some consideration was made when choosing a linear solver. The LHS matrix (onward refered to as "$\mathbb{A}$") is strictly dense, symmetric and - for a proper mass-to-stiffness ratio - positive-definite (see discussion in section 4.2.9). Due to the density, straightforward application of a Preconditioned Conjugate Gradient (PCG, see e.g.[22]) solver is not necessarily the best approach, as opposed to sparse cases such as Possion's equation. In fact, many texts recommend the use of direct solvers for the dense case. However, as we showed in 4.2.9 the system at hand has some tendencies towards poor conditioning, so robustness was a key factor in deciding on a method.

Contrary to our assumptions, the Cholesky decomposition (LL) solver didn't outperform the PCG solver by any significant means, and being a direct method it breaks down very disgracefully when the matrix becomes poorly conditioned [20]. While performance is a key issue, robustness outweighs it and so PCG should generally be prefered over LL.

The other competitor, Singular Value Decomposition (SVD), is perhaps the definition of robustness among solvers. It is a fundamental part of Principal Component Analysis (PCA), in which outlying components - corresponding to small or negative eigenvalues - can easily be filtered out [20]. The way SVD filters the solution differs from PCG though, and it turns out that the excessive damping causes visual quality to suffer. This was a totally unexpected result, and while a rigorous mathematical treatment is outside of the scope of this thesis, our take is that the damping in PCA occurs in a global fashion, which just so happens to remove many of the modes that make the motion of the Super-Helix visibly interesting. Artifacts include spurious speed decreases in all or parts of the Super-Helix, and curls and bends that do not dissolve even under the influence of high-velocity motion.

While the behaviour of standard PCG is undefined for indefinite systems, we noticed that in the general case its degenerate behaviour was visibly more pleasing than SVD. Since SVD is also slightly slower, we saw no reason to prefer it over PCG in the standard case, but maintained an option to switch to SVD to easily print out or analyze condition numbers for override purposes.

5.2 Standalone library

The standalone library provides two principal C++ classes: `SuperHelixModel` and `SuperHelixDynamics`, and the utility class `SuperHelixUtil`.

5.2.1 SuperHelixModel

At the core of the library is the SuperHelixModel class, which holds all essential information on the kinematics of the Super-Helix, i.e. curvatures, initial position and frame of reference. It also provides functions for reconstructing the Super-Helix given the kinematics parameters, with the option of providing a callback functor for discrete sampling of the curve. Spatial integration is performed symbolically using straightforward application of equation 4.22.

5.2.2 SuperHelixDynamics

`SuperHelixDynamics` deals with all dynamics. It provides an interface for adding and removing `SuperHelixModel`s to the dynamics computations, and each time its member function `integrate` is called it performs an evolution in time on all Super-Helices. It starts off by building the mass- and stiffness matrices, and assembles and computes the RHS force vectors using both symbolic and numeric integration. The $\mathbb{A}\cdot\mathbf{x} = \mathbf{b}$ system is then defined, as presented in section 4.2.8, as:

$$\mathbb{A} = \mathbb{M} + \Delta t\mu\mathbb{K} + \Delta t^2\mathbb{K} \tag{5.1}$$

$$\mathbf{b} = \Delta t\big[\mathbb{M}\dot{\mathbf{q}} + \Delta t\big([\mathbf{Q}^{int} + \mathbf{A}] + \mathbb{K}(\mathbf{q}_0 - \mathbf{q})\big)\big] \tag{5.2}$$

and the user-specified linear solver is employed to solve for $\delta\mathbf{q} = \mathbf{x}$. The resulting $\delta\mathbf{q}$ would typically be used for advance collision detection in whatever collision management will be added at a later time. Since `SuperHelixDynamics` holds information on all Super-Helices currently active in the system, it will easily be able to access that information to perform inter-collision checks and corrections.

5.2.3 SuperHelixUtil

The `SuperHelixUtil` utility class contains various utility functionality for the Super-Helix implementation, most prominently perhaps the `endpointTranslation`

method which performs the translation of endpoints described in section 4.3.

5.3 Maya plugin

As Digital Domain uses Maya for most character work we decided to create a Maya plugin to provide the Super-Helix functionality to the artists.

5.3.1 HelixShape

The HelixShape node is essentially a wrapper around the `SuperHelixModel` and `SuperHelixUtil` classes (see 5.2), with additional integration done to enable interaction with the modeling environment. It is based on the Maya API's `MPxComponentShape` class, and uses the spatial sampling hook provided by the `SuperHelixModel` to draw the Super-Helix in the OpenGL drawing override. It also uses the component facilities to give direct access to each segment of the underlying `SuperHelixModel`. Specifically, the user can pick segments by selecting their assigned vertices, and directly manipulate their curvatures and lengths using the rotation and scale tools, respectively. The requested transforms are parsed in the overriden `transformUsing` method, by extracting the rotations and scales from the homogeneous 4×4 transform matrix which is provided by the API.

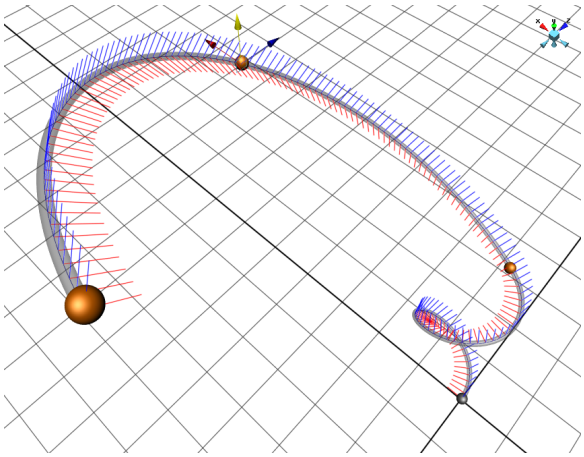

Figure 5.2: The translate tool located at one of the segment interfaces. Any user manipulation is channeled to the `SuperHelixUtil` optimization methods for endpoint translation.

To examplify the use of our novel modeling results presented in section 4.3, we also extract the translation component of the `transformUsing` matrix and use it to request segment endpoint translations from the `SuperHelixUtil`

class. This enables the artist to use the translate tool to directly translate the control vertices of the Super-Helix in a seemingly direct fashion (see Figure 5.2). Under the hood the non-linear minimization techniques of `translateEndpoint` are employed though, providing a bridge between Cartesian and generalized coordinates.

5.3.2 HelixSolver

The `HelixSolver` node contains the `SuperHelixDynamics` implementation, and accepts connections from one or multiple hair strands for input to the dynamics simulation. The node exposes a number of global dynamics parameters, such as air drag, damping/friction, thickness etc. that can be keyframed and controlled with curves like any Maya parameters.

`HelixSolver` publicly inherits `MPxNode`, the base Directed Acyclic Graph (DAG) node class. The information flow through the Maya DAG goes solely through *plugs*, which connect *attributes*. Being Acyclic, cyclic dependencies are not allowed in the DAG, since they will cause infinite loops. A main concept in the DAG is the "dirtying" and "cleaning" of attributes and plugs. Only if an attribute is dirty will the `compute` function be called upon it, so any branch of the tree that doesn't depend on the information that is updated (e.g. time) will not be evaluated but have its values read from cache instead.

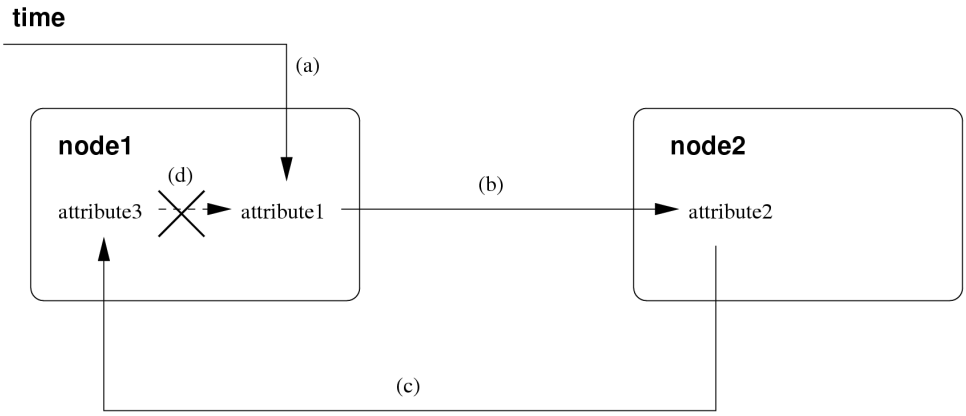

Figure 5.3: The "dirtying" flow through two DAG nodes. Time (a) is changed by the user, dirtying attribute1, which through its plug (b) sets the state of attribute2 to dirty. The plug (c) back to attribute3 is valid as long as there exists no dependence (d) of attribute1 on attribute3.

The specifics can be found in the Maya API documentation, but the significance of this property in our case is that in order for the `HelixShape` to act both as input and output to the `HelixSolver`, we had to cheat

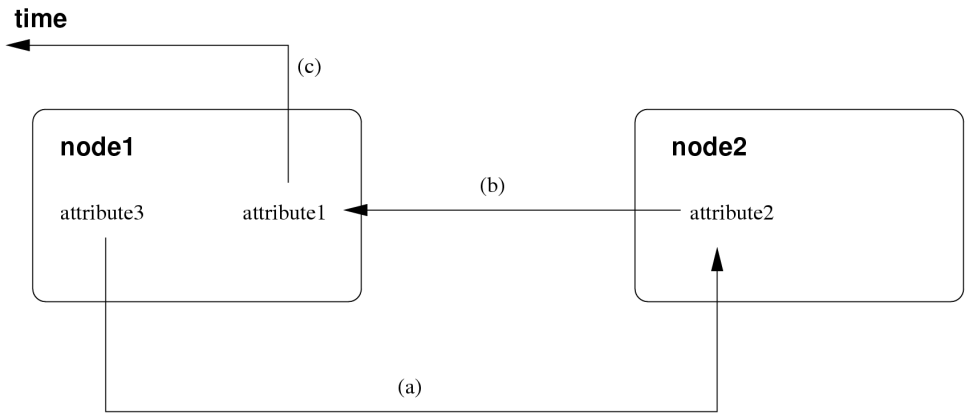

Figure 5.4: With the attributes being dirtied in figure 5.3, the subsequent `compute` calls are executed (recursively) in the order (a)-(c).

the system. Only having one data object for each Super-Helix would infallably have caused cyclic dependencies, so we also created the "proxy" attribute `curvature` which for each `HelixShape` holds an array of homogeneous point values corresponding to the three curvatures and length of each segment. This proxy attribute is used for channeling the current state of the `SuperHelixDynamics`' internal `SuperHelixModel`s to the `HelixShape` nodes, which for each *positive* timestep insert them into their internal `SuperHelixModel` objects for drawing and other user interface operations.

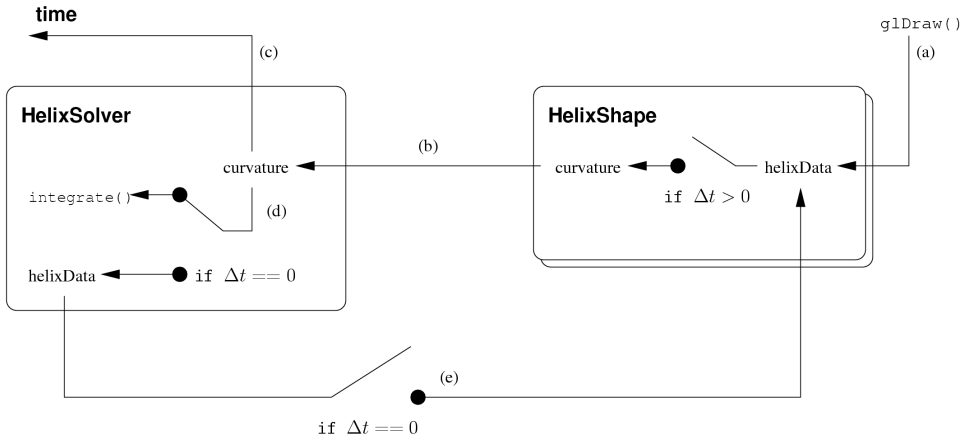

Figure 5.5: Timestep checks are used to regulate information flow through the DAG, in order to avoid cycles. Only when a positive timestep is performed is the integrator invoked.

However, whatever modifications the user does to the `HelixShape` node - which is the only visible representation of the Super-Helix - are overwritten in subsequent time steps. Thus, a pair of extra commands, `ddSetInitial` and `ddSetNatural`, were exposed, allowing the user to force the solver to read the `SuperHelixModel` state directly from the shape and set its internal representations accordingly. When any of these calls are issued though, a new dirtying of attributes - and consecutive `compute` recursions - will be (forcefully) performed, leading to cycles. As a remedy we added a number of switches to the evaluation routines, directing the information through the appropriate channels depending on if the timestep is non-zero or not (see figure 5.5). In this way we mimicked the behaviour of most dynamics tools; the user is able to at any point during simulation specify that the current state is the initial or rest state.

5.4 Grooming in Maya

With a near-complete basic setup of Super-Helix hair and dynamics, we now describe a few aspects of grooming that leverage and/or utilize the new technology.

5.4.1 Hair placement

In order to streamline the integration into the modeling environment, we used Maya Hair's own `follicle` nodes as roots for the Super-Helices (see Figure 5.6). The follicles provide an easy, efficient, way to access position and normal on a surface based on UV coordinates. While the basic interface only allows for direct selection of UV coordinates on mesh vertices, any UV coordinates can potentially be used. To facilitate this we added a noise function which randomly offsets the follicles in UV space, avoiding the "Manhattan" appearance that otherwise would frequently occur.

While being far superior in performance to the `PointOnSurfaceInfo` node for surface binding and normals, the follicles seem to be somewhat of a bottleneck in terms of scalability in the amount of Super-Helices. Already at a few hundred hairs, any translation or rotation imposed by e.g. keyframe animating the head upon which the follicles sit will cause the framerate to drop considerably. Whether this is an implementation flaw or a consequence of some of the (superfluous) functionality that the follicles provide is unknown. It seems apparent that the issue lies with the follicles though, since using other means of translation and/or rotating a few hundred Super-Helices doesn't cause a similar framerate drop. We left this issue open for future investigation.

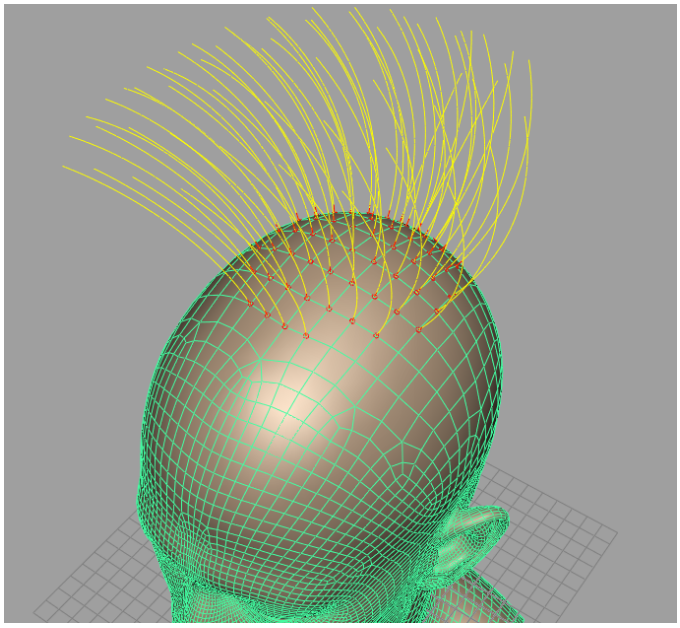

Figure 5.6: Maya Hair follicles (red) with Super-Helix hairs (yellow) attached. Without *UV* noise, the follicles' dependency on vertex positions often cause for an unnatural placement. Model courtesy of Digital Domain.

5.4.2 Combing

A strong attraction in building an internal hair dynamics engine was to aid the conception of new grooming tools, especially physically based ones. The most popular one might be the virtual comb [18],[23],[17]. Thanks to our extensive work to integrate the Super-Helix functionality into the Maya DAG framework, the Super-Helices run in Maya's *interactive playback* mode without further ado. Interactive playback enables most controls and interactions while the simulation is running, so implementing a virtual comb to work in this setting does seem very feasible to the author. The artist could for example work with a stylus pen to control the comb while using their free hand to manipulate the position and orientation of the model, e.g. by means of a 3D-mouse or a keyboard. The challenge that remains is to implement a robust collision library and put some work towards increasing the efficiency of the simulation. Some of the stylus work by Malik [15] and the multi-resolution modeling by Kim and Neumann [13] could also be of interest in this scenario.

Chapter 6

Discussion

We described an existing feature film hair pipeline, and presented the full derivations and implementation behind a new hair dynamics engine. The Super-Helix model was used, which presented a pletora of challenges while solving several problems in the previous hair solution. Here we summarize and discuss the results of the work.

6.1 Results

This section summarizes and discusses the results of the project, with focus on the positive aspects.

6.1.1 Dynamics

The motion of the Super-Helix is visually appealing even at low spatial resolutions. With only a few segments fairly complex, curly hairshapes are simulated with impressive results. The Super-Helix model has some remarkable properties; by utilizing generalized coordinates and Lagrangian mechanics, one obtains a linear system of equations describing fairly small, linear changes in curvature space that map to large, non-linear changes in Cartesian space.

The performance is roughly comparable to Maya hair without inter-hair collision at a few elements per strand, and scales equivalently. Due to the density of the system matrix, the complexity is at least $O(N^3)$ in the number of segments though, so the resolution needs to be limited. A tentative collision handling implementation suggested robust collision handling is plausible.

6.1.2 User interface

A Maya implementation was presented, in the form of a shape node and a solver node, to which several shape nodes can connect to partake in the dynamics simulation. The user interface is streamlined with other Maya tools, although still somewhat limited in flexibility due to time constraints. Efforts were made to simplify the basic process of direct modeling, in particular by using gradients to solve for curvature configurations based on user input in Cartesian coordinates. The Maya API enforces modularity and extendability, which suggests support for future extensions to the plugin.

6.2 Limitations

While the Super-Helix model certainly proved to have its strong points, it has a number of serious limitations that we present here.

6.2.1 Stiffness

The main drawback of the Super-Helix model, as it currently stands, is the issue with unproportionate hair stiffness due to the conditioning of the final discretized system. As shown in section 4.2.9, the system matrix $\mathbb{A}$, given by

$$\mathbb{A} = \rho S\mathbb{M} + (\mu\varepsilon + \varepsilon^2)\mathbb{K} \tag{6.1}$$

is positive-definite only under certain conditions. Specifically, it needs to be sufficiently diagonally dominant, a property governed by the balance between mass and stiffness. Since $\mathbb{M}$ is dense and rank-deficient, the term containing $\mathbb{K}$ needs to be given sufficient weight in the system in order to ensure positive eigenvalues. Thus, the higher the density ρ, the higher either the damping term μ or the cross-section area S needs to get. Increasing S to counter-balance mass seems like a contradiction given equation 6.1, but since the diagonal elements of $\mathbb{K}$ are quadratic in S (see section 4.2.5) the net result is an increase in the diagonal elements of $\mathbb{A}$. Ironically, this has the total opposite effect of the "stiffness" problem in nodal models (as discussed in section 2.3); we can make the hair infinitely stiff, while soft hair strands cause problems. Thus the elimination of degrees of freedom in hair length does not alone prove to be the magic bullet one might have hoped and expected.

6.2.2 Lack of hard constraints

While we showed that stable collision handling is possible in the Super-Helix model, hard constraints are inherently hard to implement in any model of generalized coordinates. Since many recent advances in advanced collision handling assume the feasibility of hard constraints, this deficit might prove

to be an insurmountable obstacle when developing a production-level collision system.

6.2.3 Unintuitive parameter space

The gains in realism made by constraining the length of the Super-Helix come at a great cost: it is very hard to facilitate intuitive manipulation of its geometry, for example in a graphical modeling environment. Such a simple task as selecting a control point somewhere along the curve and translating it, leaving the rest of the Super-Helix unmodified, is complicated by the sequential nature of the model. Optimization techniques as those presented in this thesis are required to solve the problem, at the cost of robustness and efficiency.

6.2.4 Implementation complexity

Another limitation of the Super-Helix model is the complexity of the mathematics and implementation, combined with the sparsity of previous work and lack of documentation in general. While most established hair dynamics models have seen extensive field use over the years and thus have been iterated over and discussed ad infinitum, the implementation specifics of the Super-Helix have basically yet to be filtered through the collective mind of the CG community. The symbolic integration, which lends so much stability and efficiency to the model, also introduces complexity, as the code exported from Mathematica is nearly impossible to interpret and debug. One example of this complexity is the uncertainty in the treatment of second order terms; we chose to drop them in our implementation, but more work could certainly have been done verifying this decision were the implementation any simpler and less error prone to perform.

6.2.5 Degenerate cases

On the note of complexity, the degenerate case of $\mathbf{\Omega} = \mathbf{0}$ is another complicating factor. As described in section 5.1.1, a completely straight, untwisted hair strand will cause the Darboux vector to vanish, creating havoc in the symbolic integral routines as nearly all of them have powers of the norm $|\mathbf{\Omega}|$ in the denominator. Due to time constraints we chose to settle for switching to double precision and trusting that fate prohibits the singularity from ever occuring in practice. We saw a couple of examples during testing however, mainly visible as spurious twitching and abrupt losses of kinetic energy, both of whom are obviously unacceptable in a production environment.

6.2.6 Management complexity

A problem that surfaced late in the project is the complexity associated with managing hundreds or thousands of guide hairs in a modeling envi-

ronment such as Maya. Already at a few hundred Super-Helices the Maya GUI shows significant performance problems, with the Hypergraph and similar views becoming frustatingly sluggish, and the native `follicle` nodes consuming more than half of the processor cycles. Furthermore, the scene hierarchy becomes quite hard to overview with hundreds of objects on the same leaf level. While this would be a drawback in any naive strands-based hair implementation, it certainly strengthens the case for efforts towards either researching completely different models, or implementing accelerator structures and new user tools to work around Maya's limitations.

6.3 Conclusion

The goal of the thesis project was to derive, implement and examine the complete dynamics of the Super-Helix model, as well as investigate the modeling possibilities. Also, a software implementation open to extension and further research was to be provided.

As chapter 4 shows, the derivations were performed in a rigorous and thorough manner. Chapter 5 presents the implementation which consists of a standalone library and a Maya plugin, both of which are sufficiently flexible and modular to fulfil the goal of the thesis. The modeling efforts were presented in the shape of a novel approach to directly manipulating the Super-Helix segments, bridging the gap between Cartesian and generalized coordinates.

The focus of the thesis shifted significantly during the course of the project. Very early on it became evident that the core mathematics and related implementation details would take on a much more prominent role than originally expected, at the expense of collision handling, clumping/interpolation and integration into the production pipeline. Yet the thesis fulfils the original requirements, although with a slightly different distribution of focus.

The main contributions of the thesis project are the extensive mathematics derivations and the novel modeling approach. Other contributions include discussions and insight into the conditioning of the system, as well as a complete implementation of the dynamics as a fully interactive Maya plugin.

6.4 Future work

While fulfilling its outset goals, this thesis also revealed numerous possible improvements and extensions. We present a few of them here.

6.4.1 Dynamics implementation

The stiffness problems described in sections 4.2.9 and 6.2.1 warrant further investigations before the Super-Helix model can be viewed as completely robust. Extensive tests of second order terms, and an investigation of the conditioning of the LSH matrix of the system are both crucial to solidify the understanding of the system. [2] hints at an affine relation between accelerations in Cartesian coordinates and curvatures, and refers to a book by Audoly and Pomeau titled "Elasticity and Geometry: from hair curls to the nonlinear response of shells", which may or may not have been published.

Also, the degenerate case described in section 6.2.5 needs to be dealt with appropriately, for example using polynomial interpolation over the singularity region. This is likely a very tedious task, but absolutely necessary to guarantee stability in the entire parameter space.

6.4.2 Collision handling

We presented a basic collision detection and response algorithm in section 4.2.10. Obviously there is lots of room for improvement; implementing efficient inter-hair collision detection using generalized cylinders (as suggested in [2]) would be a good first step. The collision response could either use the quadratic reduction technique mentioned in section 4.2.10 or perhaps the varying restitution coefficient as proposed for rigid bodies by Guendelman *et al.* [8]. Work could also be done to implement an adaptive timestep size, for example using backtracking. Bertails uses semi-implicit Newton with a fixed timestep, but for any penalty-based collision scheme the negative impact of the high temporal resolution required to capture high-speed interactions would probably diminish if periods of no contact were resolved at lower resolution.

6.4.3 Alternate integration schemes

Implicit and semi-implicit schemes are good for guaranteed convergence to *a* solution, which however doesn't need to be the correct one. Also, large timesteps cause excessive damping, with symptoms including persistent bends and twists that disregard interactions and gravity [25]. Investigating other integration schemes, such as high-order explicit schemes, symplectic integrators (due to the relation to Hamiltonian mechanics) [10], or even simulated annealing (to work around the conditioning problems) [20], could prove worthwile.

6.4.4 Hair management

As discussed in section 6.2.6, already at a couple of hundred hairs the Maya GUI becomes slow, unresponsive and almost unmanageable in several ways.

Both the curve editor and the Hypergraph take several minutes to update on a 3GHz Pentium 4 running only core services and Maya, and the amount of nodes makes the whole scene hard to overview and manage (see Figure 6.1). Tools for managing big sets of hair (meaning at least several thousand guide hairs) could be devised to leverage domain-specific features such as spatial coherence, clumping etc. Storing all the hairs internally in one node could improve efficiency but would also limit modularity and compatibility with other Maya components.

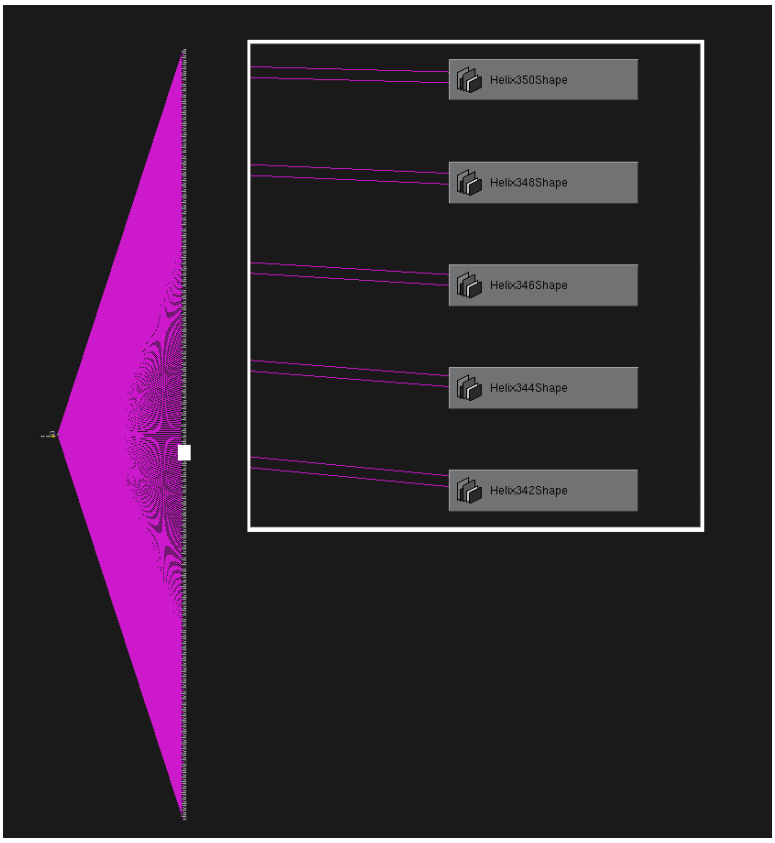

Figure 6.1: Managing thousands of guide hairs in Maya can be an overwhelming experience. On the left is the `HelixSolver` node, on the right are some four hundred `HelixShape` nodes.

6.4.5 Other models

Given the complexity and debugging difficulty of the symbolic integrals, the lack of hard constraints and the conditioning issues, it seems well-advised to test a few other recent models before settling for the Super-Helix. Op-

tions include rigid multi-body serial chains using Lagrange multipliers and hybrid methods using implicit integration. Transparently providing a fall-back method in case the Super-Helix model fails or using different models for different hair types are two options.

However, were one to address the more pressing issues discussed here, the Super-Helix model is a relatively elegant and robust take at hair dynamics simulations, with a solid foundation in classical mechanical engineering. It represents a breath of fresh air into the CG community at large and is likely to see numerous applications and extensions over the upcoming years, not only in hair modeling and dynamics.

Bibliography

[1] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics*, 26(2):303–308, 1992.

[2] F. Bertails. *Simulation de Chevelures Virtuelles*. PhD thesis, Institut National Polytechnique de Grenoble, 2006.

[3] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, and J.-L. Lévêque. Super-Helices for predicting the dynamics of natural hair. In *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, August 2006. accepted to Siggraph'06.

[4] F. Bertails, B. Audoly, B. Querleux, F. Leroy, J.-L. Lévêque, and M.-P. Cani. Predicting natural hair shapes by solving the statics of flexible rods. In J. Dingliana and F. Ganovelli, editors, *Eurographics (short papers)*. Eurographics, August 2005.

[5] F. Bertails, T.-Y. Kim, M.-P. Cani, and U. Neumann. Adaptive Wisp Tree - a multiresolution control structure for simulating dynamic clustering in hair motion. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*, July 2003.

[6] J. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. *ACM SIGGRAPH Symposium on Computer Animation*, pages 73–80, July 2002.

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

[8] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. In *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, pages 871–878, 2003.

[9] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3), 2001.

[10] E. Hairer, C. Lubich, and G. Wanner. Geometric numerical integration illustrated by the Störmer–Verlet method. *Acta Numerica, 12*, 2003.

[11] K. ichi Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In J. J. Thomas, editor, *Proceedings of the 19st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1992*, pages 111–120. ACM, 1992.

[12] R. Kameny. Software Developer, Digital Domain. *Private communication*, 2007.

[13] T. Kim and U. Neumann. Interactive multiresolution hair modeling and editing. In *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, pages 620–629, July 2002.

[14] T.-Y. Kim and U. Neumann. A thin shell volume for modeling human hair. In *CA*, pages 104–111, 2000.

[15] S. Malik. Sketching interface for modeling and editing hairstyles. In *Proceedings of Eurographics Workshop on Sketch Based Interfaces and Modeling (EGSBM)*, pages 185–194, 2005.

[16] D. Pai. Strands: Interactive simulation of thin solids using cosserat models. In *Computer Graphics Forum. Proceedings of Eurographics'02.*, pages 347–52, 2002.

[17] P. G. Palop. Technical Director Digital Domain. *Private communication*, 2007.

[18] D. Patterson. Character Artist, Digital Domain. *Private communication*, 2007.

[19] E. Plante, M.-P. Cani, and P. Poulin. A layered wisp model for simulating interactions inside long hair. In *Computer Animation and Simulation '01*, pages 139–148, 2001.

[20] W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing.* Cambridge University Press, 2002.

[21] R. E. Rosenblum, W. E. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: modeling, rendering and animation. *The Journal of Visualization and Computer Animation 2*, pages 141–48, 1991.

[22] J. R. Shewchuk. An introduction to the Conjugate Gradient method without the agonizing pain. Technical report, School of Computer Science, 1994.

[23] S. Sretschinsky. Visual Effects Supervisor, Digital Domain. *Private communication*, 2007.

[24] B. Svetitsky. Notes on functionals. Technical report, School of Physics and Astronomy, 1998.

[25] P. Volino and N. M. Thalmann. Implementing fast cloth simulation with collision response. In *CGI '00: Proceedings of the International Conference on Computer Graphics*, page 257, Washington, DC, USA, 2000. IEEE Computer Society.

[26] K. Ward, F. Bertails, T.-Y. Kim, S. R. Marschner, M.-P. Cani, and M. Lin. A survey on hair modeling: Styling, simulation, and rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 13(2):213–234, Mar-Apr 2007. To appear.

[27] K. Ward, M. Lin, J. Lee, S. Fisher, and D. Macri. Modeling hair using level-of-detail representations. *International Conference on Computer Animation and Social Agents, May 2003*, 2003.

[28] Y. Wei, E. Ofek, L. Quan, and H.-Y. Shum. Implementation of modeling hair from multiple views. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 149, New York, NY, USA, 2005. ACM Press.

[29] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. In F. Baskett, editor, *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, pages 243–252, 1990.

[30] Y. Yu. Modeling realistic virtual hairstyles. In *Proceedings of Pacific Graphics*, pages 295–304, 2001.