STRANDS AND HAIR

MODELING, ANIMATION, AND RENDERING

SIGGRAPH 2007 Course Notes

May 2, 2007

Course Organizer: Sunil Hadap

Presenters:

Sunil Hadap
Adobe Systems[1]
sunilhadap@acm.org

Marie-Paule Cani
INP Grenoble / INRIA
marie-paule.cani@imag.fr

Ming Lin
University of North Carolina
lin@cs.unc.edu

Tae-Yong Kim
Rhythm&Hues Studio
tae@rhythm.com

Florence Bertails
University of British Columbia
bertails@cs.ubc.ca

Steve Marschner
Cornell University
srm@cs.cornell.edu

Kelly Ward
Walt Disney Animation Studios
kelly.ward@disney.com

Zoran Kačić-Alesić
Industrial Light+Magic
zoran@ilm.com

[1]formerly at PDI/DreamWorks

Contents

Introduction

Course Abstract

The last six years has seen a renaissance in hair modeling, rendering and animation. This course covers the gamut of hair simulation problems and present working solutions, from recent and novel research ideas to time tested industrial practices that resulted in spectacular imagery.

The course is aimed at an intermediate level and addresses the special-effects developers and technical directors who are looking for innovation as well as proven methodologies in hair simulation. The audience will get a good grasp of the state of the art in hair simulation and will have plenty of working solutions that they can readily implement in their production pipelines. The course will also be a bootcamp for aspiring computer graphics researchers interested in physically based modeling in computer graphics.

The course reviews all the three main tasks in hair simulation: hairstyling, hair animation and hair rendering. First, the nature of human hair will be shortly discussed, to point out why modeling this material has been such a challenge for Computer Graphics. The basics for coping with the hair complexity oin terms of the number of strands will be given. Subsequently, the latest methodologies pertaining to each of the tasks will be covered. For hair dynamics, we will cover an elaborate as well as viable stiffness dynamics model of an individual hair strand. We will present two novel models for complex hair-hair, hair-body and hair-air interactions. For rendering, we discuss issues related to shading models, multiple scattering, and volumetric shadows. We also present the recent advances in programmable graphics hardware in the context of hair rendering. Advanced techniques in production quality hair rendering, self shadowing and lighting models

for short and long hair will also be presented.

Prerequisites: Familiarity with fundamentals of computer graphics, physical simulation and physically based rendering is strongly recommended but not mandatory. Understanding of numerical linear algebra, differential equations, numerical methods, rigid-body dynamics, collision detection and response, physics-based illumination models, fluid-dynamics would be a plus.

Target audiences include special effects developers, technical directors, game developers, researchers, or any one interested in physically based modeling for computer graphics.

Presenters Biography

Sunil Hadap, Adobe Systems (formerly at PDI/DreamWorks), USA

Sunil Hadap was a member of R&D Staff at PDI/DreamWorks, developing next generation dynamics tools for use in upcoming film productions. His research interests include a wide range of physically based modeling problems such as clothes, fluids, rigid body dynamics and deformable models. Sunil developed strand and hair simulation techniques at PDI/DreamWorks. The resulting system is extensively used in the production of Madagascar, and upcoming movies Shrek The Third and Bee Movie. The methodology and the results are presented at SCA 2006. Sunil Hadap completed his PhD in Computer Science from MIRALab, University of Geneva under the guidance of Prof. Nadia Magnenat-Thalmann. His PhD thesis work was in Hair Simulation. During his doctoral research, he pioneered techniques for hairstyling and hair animation, which were published in Eurographics 2001. He recently joined the Advanced Technology Labs at Adobe Systems as a Senior Graphics Researcher.

Marie-Paule Cani, INP Grenoble and INRIA, France

Marie-Paule Cani is a professor at the Institut National Polytechnique de Grenoble, France. A graduate from the Ecole Normale Supérieure, she received a PhD

in Computer Science from the University of Paris Sud in 1990. She was awarded membership of the Institut Universitaire de France in 1999. Her main research interest is to explore the use of geometric and physically-based models for designing complex interactive scenes. Recent applications include the animation of natural scenes (lava-flows, ocean, meadows, wild animals, human hair) and interactive sculpting or sketching techniques. Marie-Paule Cani has served in the program committee of the major CG conferences. She co-chaired IEEE Shape Modelling & Applications in 2005 and was paper co-chair of EUROGRAPHICS'2004 and SCA'2006.

Ming Lin, University of North Carolina, USA

Ming Lin received her Ph.D. in EECS from the University of California, Berkeley. She is currently a Professor of Computer Science at the University of North Carolina, Chapel Hill. She received several honors and five best-paper awards. She has authored over 160 refereed publications in physically-basedmodeling, haptics, robotics, and geometric computing. She has served as the conference and program chair of many conferences and the steering committee member of ACM SIGGRAPH / Eurographics Symposium on Computer Animation. She is also an associated and guest editor of several journals and magazines. She has given many lectures at SIGGRAPH and other international conferences.

Tae-Yong Kim, Rhythm & Hues Studios, USA

Tae-Yong Kim is currently a software engineer in Rhythm and Hues Studios. His primary responsibility at R&H includes development of simulation tools such as cloth, hair, and other types of softbody. For the Chronicles of Narnia, he developed a new hair simulation technique that was used to simulate aslan's fur as well as fur and hair of other creatures. He holds a Ph.D degree in computer science from the University of Southern California where he did researches on hair modeling and rendering techniques. His work was published in SIGGRAPH 2002 as well as other conferences. He was a lecturer in 2003, 2004 SIGGRAPH courses on hair simulation. He also recently taught a SIGGRAPH course about the movie 'The Chronicles of Narnia' in 2006.

Florence Bertails University of British Columbia, Canada

Florence Bertails is currently a postdoctoral researcher at the University of British Columbia. A graduate from the Telecommunication Engineering School of INPG (France), she received in 2002 a MSc in Image, Vision and Robotics, and completed in 2006 a PhD on hair simulation at INPG. Her research interests focus on physically-based modelling and animation of passive objects, such as hair or fluids. She presented her work at international conferences such as the ACM-EG Symposium of Computer Animation, Eurographics, and Graphics Interface (best student paper award in 2005). Her latest work on hair animation was published at ACM SIGGRAPH in 2006.

Zoran Kačić-Alesić, Industrial Light & Magic, USA

Zoran Kačić-Alesić is a principal R&D engineer at Industrial Light & Magic, leading a team responsible for structural simulation and sculpting/modeling tools. His work at ILM (since 1992) has been primarily in the area of 3D modeling and animation including interactive tools for skin animation, 3D painting, sculpting, 3D scanning and model reconstruction, digital clay, choreography, and dynamics simulations, as well as software architecture for large 3D applications. His movie credits include Jurassic Park, Star Wars EP1, Harry Potter 4, and Pirates of the Caribbean 2. In 1996 he received a Scientific and Engineering Academy Award for the development of the ViewPaint 3D Paint System for film production work. Zoran received a BEng in electrical engineering from the University of Zagreb, Croatia; a MSc in computer science from the University of Calgary, Canada; and an honorary doctorate in fine arts from the University of Lethbridge, Canada.

Steve Marschner, Cornell University, USA

Steve Marschner is Assistant Professor of Computer Science at Cornell University, where he is conducting research into how optics and mechanics determine the appearance of materials. He obtained his Sc.B. from Brown University in 1993 and his Ph.D. from Cornell in 1998. He held research positions at Hewlett-Packard Labs, Microsoft Research, and Stanford University before joining Cornell in 2002. He has delivered numerous presentations, including papers at IEEE

Visualization, the Eurographics Rendering Workshop, and SIGGRAPH, and SIGGRAPH courses every year from 2000 to 2005. For contributions in rendering translucent materials, he is co-recipient with Henrik Wann Jensen and Pat Hanrahan of a 2003 Academy Award for technical achievement.

Kelly Ward, Disney Animation, USA

Kelly Ward is currently a software engineer at Walt Disney Feature Animation, where she works on hair simulation and modeling tools for feature films. She received her M.S. and Ph.D. degrees in Computer Science from the University of North Carolina, Chapel Hill in 2002 and 2005, respectively. She received a B.S. with honors in Computer Science and Physics from Trinity College in Hartford, CT in 2000, where she was named the President's Fellow in Physics in 1999-2000. Her research interests include hair modeling, physically-based simulation, and computer animation. She has given several presentations and invited lectures on her hair modeling research at international venues.

Course Syllabus

Session 1. Virtual Hair: motivations and challenges (Marie-Paule Cani)

The peculiar nature of hair will be discussed, which would enable us to introduce the challenges this material pose in terms of modeling, animation and rendering. Coping with the number of strands is the first major issue when modeling virtual hair. Towards that, the basic idea of animating fewer guide strands and adding extra ones at the rendering stage will be presented. This will enable us to introduce the main aspects discussed in this course - the need for good dynamic models for guide strands, for specific algorithms handling hair interaction and for appropriate rendering techniques.

Session 2. Dynamics of Strands

Oriented Strands – a versatile dynamic primitive (Sunil Hadap)

The simulation of strand like primitives modeled as dynamics of serial branched multi-body chain gives rise to stiff and highly non-linear differential equations. We introduce a recursive, linear time and fully implicit method to solve the stiff dynamical problem arising from such a multi-body system. We augment the merits of the proposed scheme by means of analytical constraints and an elaborate collision response model.

Super Helices – dynamics of thin geometry (Florence Bertails)

We introduce the mechanical model based on Super Helix. This model is defined as a piece-wise helical rod, and can represent the essential modes of deformation (bending and twisting) of a strand, as well as a complex rest geometry (straight, wavy, curly) in a very compact form. We develop the kinematics of the model, as we derive the dynamic equations from the Lagrange equations of motion. Finally, we provide a rigorous validation for the Super Helix model by comparing its behavior against experiments performed on real hair wisps.

Session 3: Hair Interaction

Coping with the complexity of hair-interactions – hair density, adaptive wisp-tree and temporal coherence (Marie-Paule Cani)

Being able to take hair self-interactions into account is essential for bridging the gap between the simulation of isolated strands and the animation of a full head of hair. We discuss three solutions for coping with the complexity of hair interactions, belonging either to a continuum or to a wisp-based vision of hair: After presenting a real-time, volumetric solution based on hair density, we explain how hair can alternatively be seen as as an adaptive tree of wisps refining into sub-wisps at the tip and explain how temporal coherence can be accounted for when processing hair self-interactions.

Multi-resolution hair-hair and hair-obstacle interaction (Ming Lin, Kelly Ward)

We present novel geometric representations, simulation techniques, and numerical methods to significantly improve the performance of hair dynamics computation, hair-object and hair-hair interactions. These approaches focus on balancing visual fidelity and performance to achieve realistic appearance of animated hair at interactive rates. In addition, we discuss application and system requirements that govern the selection of appropriate techniques for interactive hair modeling.

Session 4 : Hair Rendering (Steve Marchner and Tae-Yong Kim)

In this session, we cover the state-of-the-art in hair rendering. We start with a comprehensive yet practical theory behind physically based hair rendering, including light scattering through hair volume and self shadowing. We then present innovative techniques to speed-up hair rendering tasks using graphics hardware.

Session 5 : Simulation of Strands and Hair in Feature Productions

Madagascar & Shrek The Third - strands and foliage ears, braid, foliage and long hair (Sunil Hadap)

We discuss the use of a versatile simulation system based on the strand primitive, introduced in seesion 2, for character dynamics and visual effects in Madagascar and Shrek The Third. We demonstrate the dynamics of ears, braid, long/curly hair and foliage.

Hair and strand simulation at ILM (Zoran Kačić-Alesić)

We first give an overview of the hair and strand simulation techniques used in a production environment at ILM. Then we draw upon examples of highly dynamic long hair (Vampire Brides in "Van Helsing"), full body medium length fur (werewolves, wolves, Wookies in "Van Helsing", "The Day After Tomorrow", and "Star Wars Episode 3"), and articulated tentacles simulations (Davy Jones in "Pirates of the Caribbean 2"). We will explore commonalities between hair, cloth, and rigid body simulations, along with situations in which they can be used together or interchangeably.

Chronicles of Narnia – simulation of lion mane and other hair fur techniques (Tae-Yong Kim)

Session 6 : Questions and Discussions

Chapter 1

Dynamics of Strands

Sunil Hadap, Florence Bertails, Basile Audoly, Marie-Paule Cani

1.1 Introduction

Realistic hair simulation is one of the most difficult issues when animating virtual humans. Human hair is a very complex material, consisting of hundreds of thousands of very thin, inextensible strands that interact with each other and with the body. Unlike solids or fluids, which have been studied for over a century and well modeled by now classical equations, hair remains a largely unsolved problem described by no well accepted model. Finding a representation that provides an accurate simulation of hair motion remains a challenge.

Modeling hair dynamics raises a number of difficulties. The very first one is due to the fact that each individual strand has a complex nonlinear mechanical behavior, strongly related to the thinness of its cross section as well as its natural shape: smooth, wavy, curly or fuzzy. In this chapter, after a brief report about the mechanical structure and properties of hair strands, we present two innovative models that allow to capture the main dynamic features of thin geometry. The first model, called *Oriented Strands*, is based on a stable integration of serial body dynamics, and nicely incorporates external constraints. The second model, called *Super-Helices*, provides a compact high-order representation for a strand of arbitrary geometry (straight, wavy, curly), and captures the essential modes of

deformation of a real fiber; this model was strongly validated against experiments made on real hair.

The difficult problem of interactions and contacts to be accounted for when simulating a full head of hair, and the question of efficiency, will be adressed in chapter 2 and chapter 3, respectively.

1.2 Hair structure and mechanics

Achieving realistic simulations of hair motion requires some understanding of hair structure. This section gives a summary of the existing knowledge on individual hair strands, mostly issued from the field of cosmetics. Further details can be found in [LFHK88, Rob02].

A human hair fiber is a thin structure (about 0.1 mm in diameter) with either a circular or oval cross section. The active part, called the *follicle*, is located under the skin and produces the keratin proteins that compose the hair material. The second part, the visible – and dead – part of hair, is called the *hair shaft*, and corresponds to the "hair strand" we are seeking to animate.

The hair shaft is entirely synthesized by the associated follicle, which acts as a mold for shaping the strand [LFHK88]. It thus has almost uniform cross section, natural twist and natural curvatures all along. These geometric parameters are associated with commonsense notions of straight, curly, or fuzzy hair. Their values are characteristic of the *ethnic group* from which the hair comes [Rob02]. Africans have follicles with a helical form and an oval cross section, whereas Asians have follicles that are completely straight with a larger and circular cross section. As a result, Asian hair is thicker, with no natural curliness. It makes it look smooth and regular. In contrast, African hair looks frizzy and irregular. Caucasian hair stands between these two extremes.

The internal structure of the shaft consists of three concentric layers from the core to the periphery: a central canal called *medulla*; the *cortex*, *i.e.* cells filled with keratin, contributing 90% of the total weight; and the *cuticle*, a thin coating covered by tilted scales. Keratin is a remarkably stiff material, making the shaft extremely difficult to *shear* and *stretch*. However, because its cross section is very small, it can be easily *bent* and *twisted*.

Figure 1.1: Left, close view of a hair fiber (root upwards) showing the cuticle covered by overlapping scales. Right, bending and twisting instabilities observed when compressing a small wisp.

Deformations of a hair strand involve rotations that are not infinitely small and so can only be described by *nonlinear* equations [AP07]. Physical effects arising from these nonlinearities include instabilities called *buckling*. For example, when a thin hair wisp is held between two hands that are brought closer to each other (see Figure 1.1, right), it reacts by bending in a direction perpendicular to the applied compression. If the hands are brought even closer, a second instability occurs and the wisp suddenly starts to coil (the bending deformation is converted into twist).

1.3 Oriented Strands: a versatile dynamic primitive

The simulation of strand like primitives modeled as dynamics of serial branched multi-body chain, albeit a potential reduced coordinate formulation, gives rise to stiff and highly non-linear differential equations. We introduce a recursive, linear time and fully implicit method to solve the stiff dynamical problem arising from such a multi-body system. We augment the merits of the proposed scheme by means of analytical constraints and an elaborate collision response model. We finally discuss a versatile simulation system based on the strand primitive for character dynamics and visual effects. We demonstrate dynamics of ears, braid, long/curly hair and foliage.

1.3.1 Introduction and Related Work

The simulation of ears, tails, braids, long wavy/curly hair, foliage, jewelry is peculiar in nature. The flexible shape is characterized by a thin and long geometry, which typically has a non-straight rest configuration. The dynamics predominantly includes the bend and the torsional components, and very rarely the length-wise stretch component. Even though being one-dimensional in nature, the intricate rendering aspects of these primitives, along with potentially highly anisotropic physical properties, demand a consistent/stable curvilinear coordinate system all along the geometry. Here, we would like to present a versatile dynamic primitive that spans the stated characteristics and applications. We name the system as Oriented Strands, to clearly convey the picularity to the user.

Cosserat Models discussed in [Rub00] and first introduced to computer graphics community by [Pai02b] give an elaborate continuum theory behind the dynamics of thin deformable objects such as strand and shells. The discrete approximation of the strand model come strikingly close to the strand-as-serial-multi-body-chain model, first proposed by [HMT01b, Had03]. Since then the paradigm is successfully used for hair simulation by [CJY02b, CCK05b]. We too model strand as serial chain of rigid segments connected by spherical joints. Previously, the hair was typically modeled using mass-spring-hinge system, as individual hair strands [RCT91a, AUK92a] or as wisps [PCP01b]. However, these models are not effective in representing consistent coordinates and the twist dynamics. An exhaustive overview of various hair simulation techniques is given in [Had03].

[Fea87] developed one of the first multi-body reduced coordinate formulations that has a linear complexity. [Mir96, Kok04] further developed efficient and comprehensive impulse and constraint formulations to it. [RGL05b] extended the formulation to achieve interesting sub-linear complexity, and also gives a thorough overview of the other reduced coordinate formulations. [Bar96, Sha01] gives maximal coordinate formulations which also are known to have linear complexity using sparse-matrix solution methods. The typical multi-body system resulting from the strand model, gives rise to "stiff" and highly non-linear differential equations. The numerical difficulties stem from small rotational inertia along the axis due to thin geometry, large bend and torsional stiffness-to-mass-ratio and intricate non-straight rest shape. The non-linearity is due to velocity terms corresponding to Coriolis forces and the specific choice of the non-linear elastic model in our implementation to limit unrealistic high deformations. These difficulties call for

an implicit integration scheme. Even though the reduced coordinate formulation is efficient for multi-body systems with large number of segments and relatively small number of DOFs, it is difficult to realize an implicit integration scheme, as pointed out by [Had03]. Instead, [CCK05b] use a traditional maximal coordinate formulation with (soft) constraints [Bar96, Sha01], followed by an implicit integration. Complex collision response models with static and dynamic friction can be integrated into the maximal coordinate framework, with relative ease, using impact and contact velocity constraints [CCK05b].

Nevertheless, the reduced coordinate formulation has certain advantages. The generalized coordinates directly facilitate the parameterization for bending and torsional stiffness dynamics. Further, they have the exact same form as the parameterization used in articulated character animation. Thus the rest shape of the multi-body system can be conveniently defined in terms of some animated local, e.g. a hairstyle can be defined in terms of the frame associated with the head joint. Even the dynamics is often expressed in terms of successive local coordinates starting from the base link. One can thus interpret the dynamic motion of the strand as overall rigid-body transformation of the strand at the base link, followed by secondary dynamics from the rest shape expressed in terms of successive local frames. This paradigm gives a tremendous advantage in terms of overall simulation workflow. Typically the base link is constrained to an animation path. Using the paradigm, it is trivial to kick-start the simulation from an arbitrary starting position and orientation of the base link. Moreover, certain concepts such as posable dynamics, local dynamics, time-scale and zero-gravity rest shape make the strand simulation system versatile. As discussed subsequently, they are often trivial to realize in the paradigm of reduced coordinate formulation. Ultimately, the choice of reduced coordinate formulation proved very rewarding for us.

The specific contributions of the methodology are as follows. In Section 1.3.2 and Section 1.3.3 we develop a linear time, implicit and fully recursive scheme for reduced coordinate formulation of general branched open-chain multi-body system, using Differential Algebraic Equations (DAE). In Section 1.3.4, we discuss how to realize external bilateral and unilateral constraints on the formulated multi-body dynamics. We also discuss the numerical issues associated with the solution of Linear Complementarity Problem (LCP) arising from the formulation. In Section 1.3.5, we develop an elaborate collision response model with inelastic impact and static/dynamic friction, using unilateral constraints. Finally, in Section 1.3.6, we introduce the Oriented Strands system, implemented as dynamics of serial multi-body chain. We develop some novel concepts and give important

implementation details that makes the dynamic strand primitive versatile. In Section 5.1, we present some illustrative examples of dynamics of ears, braid, hair and foliage.

1.3.2 Differential Algebraic Equations

Typically, unconstrained dynamical problems such as cloth [BW98b] and general deformable models are formulated as the following *explicit* Ordinary Differential Equation (ODE) of degree two.

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{Q}(t, \mathbf{q}, \dot{\mathbf{q}}) \tag{1.1}$$

Constrained dynamical problems such as dynamics of multi-body systems [Sha01, Bar96] are formulated as the following *semi-explicit* ODE of degree two.

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} &= \mathbf{Q}(t, \mathbf{q}, \dot{\mathbf{q}}) - {\Phi_{\mathbf{q}}}^T \lambda \\ \Phi(t, \mathbf{q}) &= 0 \end{aligned} \tag{1.2}$$

where, $\mathbf{M}$ is generalized mass matrix. The force function $\mathbf{Q}$ and the constraint function Φ are typically non-linear and "stiff". In order to integrate the state vector $[\dot{\mathbf{q}}^T, \mathbf{q}^T]_t^T$, in a traditional way, one can try and solve for the derivatives of the state vector $[\ddot{\mathbf{q}}^T, \dot{\mathbf{q}}^T]_{t+1}^T$, which often turns out to be complex. Fortunately, the direct computation of derivatives is not the only way, neither it is the most efficient way, of solving the differential equations. Differential Algebraic Equations solvers are remarkable, they advance the solution $[\dot{\mathbf{q}}^T, \mathbf{q}^T]_t^T \rightarrow [\dot{\mathbf{q}}^T, \mathbf{q}^T]_{t+1}^T$, as they estimate the derivatives $[\ddot{\mathbf{q}}^T, \dot{\mathbf{q}}^T]_{t+1}^T$ at the same time.

As far as we can track, Differential Algebraic Equations (DAE) are new to computer graphics. In this section we would like to give a gentle introduction to DAE. For thorough discussion and associated theory we would like to refer to [BCP96]. DAE solvers work on the system of differential equations in its most implicit form. Consider the following DAE of degree one.

$$\mathbf{F}(\mathbf{y}, \dot{\mathbf{y}}, t) = \mathbf{0} \tag{1.3}$$

The implicit function $\mathbf{F}$ in differential variables $\mathbf{y}$ and free variable t may be non-linear and "stiff". Let the set $\{\mathbf{y}, \dot{\mathbf{y}}\}_t$ be the solution of the DAE, i.e. it satisfies

equation 1.3 at time t. Then the DAE solvers use an extrapolation method, e.g. Backward Difference Formula (BDF) of an appropriate order, to extrapolate the solution to $\mathbf{y}^1_{t+1}$ and while making a numerical estimate of the derivative $\dot{\mathbf{y}}^1_{t+1}$. The estimates $\mathbf{y}^1_{t+1}, \dot{\mathbf{y}}^1_{t+1}$ typically would not satisfy equation 1.3. The solver then successively corrects the solution and associated derivative estimate by number of Newton-Raphson iterations. Let the residue associated with the estimate of k^{th} iteration be

$$\mathbf{rs}^k_{t+1} = \mathbf{F}(\mathbf{y}^k_{t+1}, \dot{\mathbf{y}}^k_{t+1}, t+1) \tag{1.4}$$

The Newton-Rapson iteration takes the following form

$$\begin{aligned} \mathbf{y}^{k+1}_{t+1} &= \mathbf{y}^k_{t+1} - \frac{\partial \mathbf{F}}{\partial \mathbf{y}}^{-1} \mathbf{rs}^k_{t+1} \\ \dot{\mathbf{y}}^{k+1}_{t+1} &= (\mathbf{y}^{k+1}_{t+1} - \mathbf{y}_t)/\triangle t \end{aligned} \tag{1.5}$$

Thus, in order to use DAE integrator such as DASPK [BCP96], one has to provide the integrator with a residual function **rs** that computes the residue from the estimates of the solution the integrator provides. One may also provide the Jacobian of the residue function $\partial \mathbf{F}/\partial \mathbf{y}$, or optionally the integrator can compute the Jacobian numerically. The highlight of the solution method is – the residue function **rs** and the Jacobian $\partial \mathbf{F}/\partial \mathbf{y}$ are often simple to evaluate. In the next section, we formulate a fully recursive method to evaluate the residual function for solution of a "sitff" multi-body system.

1.3.3 Recursive Residual Formulation

To describe the dynamics of the multi-body system, we use Spatial Algebra and associated notation developed by [Fea87]. Consider a serial branched multi-body system (MBS) having n links connected by n single DOF joints as shown in Figure 1.2. There are no loops in the kinematic chain.

The base link is denoted by $link_0$ and is typically constrained to a motion path. The other links are numbered in a breadth first manner. The velocity $\hat{\mathbf{v}}_j$, the acceleration $\hat{\mathbf{a}}_j$ and the inertia matrix $\hat{\mathbf{I}}_j$ of $link_j$ are defined in the frame $\hat{\mathbf{F}}_j$, which is rigidly attached to the link's principal inertia axis. The joint variable q_j defines the spatial transformation $\hat{\mathbf{X}}_j$ that transforms the spatial quantities defined in the

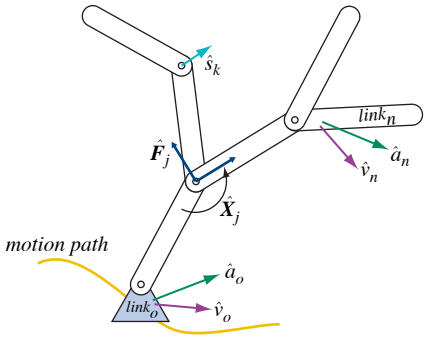

Figure 1.2: Strand as multi-body system

parent's frame $\hat{\mathbf{F}}_i$ to the frame $\hat{\mathbf{F}}_j$ of $link_j$. The derivatives of joint variables $\dot{q}_j$ and $\ddot{q}_j$ relate the velocity and acceleration of the parent to the velocity and acceleration of $link_i$ via the spatial joint axis $\hat{\mathbf{s}}_i$.

$$\begin{aligned} \hat{\mathbf{v}}_j &= \hat{\mathbf{X}}_j\, \hat{\mathbf{v}}_i + \hat{\mathbf{s}}_j\, \dot{q}_j \\ \hat{\mathbf{a}}_j &= \hat{\mathbf{X}}_j\, \hat{\mathbf{a}}_i + \hat{\mathbf{s}}_j\, \ddot{q}_j + \hat{\mathbf{v}}_j \hat{\times} \hat{\mathbf{s}}_j\, \dot{q}_j \end{aligned} \tag{1.6}$$

The set of joint variables $\mathbf{q}_t$ and their derivative $\dot{\mathbf{q}}_t$ forms the system state vector $\mathbf{y}_t = [\dot{\mathbf{q}}^T, \mathbf{q}^T]_t^T$, at time t. We would like to solve the forward dynamical problem – given the base velocity $\hat{\mathbf{v}}_0$ and the base acceleration $\hat{\mathbf{a}}_0$, integrate the state from $\mathbf{y}_t$ to $\mathbf{y}_{t+1}$. In what follows we will develop a recursive residual formulation based on DAE methodology discussed in the previous section. The discussion is rather a free physical interpretation, for the rigorous proof refer to [RJFdJ04]. The procedure is surprisingly simple as compared to the traditional methods such as Articulated Body Method [Fea87], where one computes the state derivative $\dot{\mathbf{y}}_{t+1}$ explicitly.

The solution set $\{\mathbf{y}_t, \dot{\mathbf{y}}_t\}$ at time t forms the input to the DAE integrator. As highlighted in the previous section, the integrator then estimates the new state $\mathbf{y}_{t+1}^k$, and it's derivative $\dot{\mathbf{y}}_{t+1}^k$ in succession. It is our responsibility to compute the associated residue $\mathbf{rs}_{t+1}(\mathbf{y}^k, \dot{\mathbf{y}}^k)$. Given $\hat{\mathbf{v}}_0$ and $\hat{\mathbf{a}}_0$, we first compute the spatial velocities $\hat{\mathbf{v}}_j$ and spatial accelerations $\hat{\mathbf{a}}_j$ for each link $link_j$, $j = 1..n$ using forward kinematic relation, equation 1.6.

The residue associated with accelerations can be computed using the force balance condition. Starting with the outer most link $link_n$, the forces acting on $link_n$ are

spatial body force $\hat{\mathbf{I}}_n \hat{\mathbf{a}}_n$, combined spatial centripetal and Coriolis force $\hat{\mathbf{v}}_n \hat{\times} \hat{\mathbf{I}}_n \hat{\mathbf{v}}_n$, external spatial force $\hat{\mathbf{f}}_n$. The force balance equation for the spatial forces is

$$\hat{\mathbf{rs}}_n \;=\; \hat{\mathbf{I}}_n \hat{\mathbf{a}}_n \;+\; \hat{\mathbf{v}}_n \hat{\times} \hat{\mathbf{I}}_n \hat{\mathbf{v}}_n \;-\; \hat{\mathbf{f}}_n \tag{1.7}$$

We still have to relate the force residue $\hat{\mathbf{rs}}_n$ which is a spatial vector to the residue in joint acceleration which is a scalar. We project the force residue on to the joint's motion sub-space defined by the spatial joint axis $\hat{\mathbf{s}}_n$.

$$rs_n \;=\; \hat{\mathbf{s}}_n^S \, \hat{\mathbf{rs}}_n - Q_n \tag{1.8}$$

where, $\hat{\mathbf{s}}_n^S$ is spatial transpose of the joint axis and Q_n is scalar joint actuation force associated with the stiffness model. The force residue projected on the joint's motion space rs_n vanishes when the estimated state and derivative vector is the solution of DAE.

For simplicity, first consider a multi-body system with no branches. Thus, the only parent of $link_n$ would be $link_{n-1}$. In computing the force balance equation for $link_{n-1}$, along with all the obvious forces described for $link_n$, we need to add the residual force from $link_n$ that gets applied via the $joint_n$. In order to do that, we need to transform the force residue $\hat{\mathbf{rs}}_n$ into the frame of $link_{n-1}$, using inverse transformation matrix $\hat{\mathbf{X}}_n^{-1}$. The resulting force balance equation for $link_{n-1}$ is

$$\begin{aligned} \hat{\mathbf{rs}}_{n-1} \;&=\; \hat{\mathbf{I}}_{n-1} \hat{\mathbf{a}}_{n-1} \;+\; \hat{\mathbf{v}}_{n-1} \hat{\times} \hat{\mathbf{I}}_{n-1} \hat{\mathbf{v}}_{n-1} \;-\; \hat{\mathbf{f}}_{n-1} \\ &\quad + \hat{\mathbf{X}}_n^{-1} \, \hat{\mathbf{rs}}_n \\ rs_{n-1} \;&=\; \hat{\mathbf{s}}_{n-1}^s \hat{\mathbf{rs}}_{n-1} - Q_{n-1} \end{aligned} \tag{1.9}$$

We repeat this process for each $link_i$ till we reach the first link $link_1$. The residue associated with the joint velocities is trivially the difference in joint velocities $\dot{q}_i^{k*} - \dot{q}_i^k$, where $\dot{q}_i^k$ belongs to $\mathbf{y}^k$ and $\dot{q}_i^{k*}$ belongs to $\dot{\mathbf{y}}^k$.

Algorithm 1 lists the fully-recursive algorithm for computing the residue, for a general multi-body system that have branches.

It is possible to compute the analytic Jacobians for the recursive residue formulation [RJFdJ04]. Alternatively, we can let the DAE solver compute the Jacobians numerically. We particularly commend the efficient and "smart" evaluations of Jacobians in DASPK, the DAE solver we used for the implementation. The solver

uses modified Newton-Rapson iteration, where the Jacobians are evaluated only when the solver observes a large change in the system's state. In practice, we found that the numerical evaluation of Jacobians is not only adequate, but also versatile. Thus, we can implement any complex stiffness model and associate general external fields to the multi-body system, as discussed in Section 1.3.6. It may not be possible to evaluate analytic Jacobians for these.

Algorithm 1 $res_{t+1}(\mathbf{y}^k,\ \dot{\mathbf{y}}^k, \hat{\mathbf{v}}_0, \hat{\mathbf{a}}_0)$

Require: $\mathbf{y}^k = \begin{bmatrix} \mathbf{q}^k \\ \dot{\mathbf{q}}^k \end{bmatrix}$, $\dot{\mathbf{y}}^k = \begin{bmatrix} \dot{\mathbf{q}}^{k*} \\ \ddot{\mathbf{q}}^k \end{bmatrix}$

$n \leftarrow dim(\mathbf{q})$
for $j = 1$ to n **do**
  $i \leftarrow parent(link_j)$
  $\hat{\mathbf{v}}_j \leftarrow \hat{\mathbf{X}}_j\ \hat{\mathbf{v}}_i + \hat{\mathbf{s}}_j\ \dot{q}_j$
  $\hat{\mathbf{a}}_j \leftarrow \hat{\mathbf{X}}_j\ \hat{\mathbf{a}}_i + \hat{\mathbf{s}}_j\ \ddot{q}_j + \hat{\mathbf{v}}_j \hat{\times} \hat{\mathbf{s}}_j\ \dot{q}_j$
end for

$\hat{\mathbf{rs}} \leftarrow \hat{\mathbf{0}} \in \Re^{6n}$
$\mathbf{rs} \leftarrow \mathbf{0} \in \Re^n$
for $i = n$ to 1 **do**
  $\hat{\mathbf{rs}}_i \leftarrow \hat{\mathbf{I}}_i\ \hat{\mathbf{a}}_i\ +\ \hat{\mathbf{v}}_i \hat{\times} \hat{\mathbf{I}}_i \hat{\mathbf{v}}_i\ -\ \hat{\mathbf{f}}_i$
  for all $j \leftarrow child(link_i)$ **do**
    $\hat{\mathbf{rs}}_i \leftarrow \hat{\mathbf{rs}}_i + \hat{\mathbf{X}}_j^{-1}\ \hat{\mathbf{rs}}_j$
  end for
  $rs_i \leftarrow \hat{\mathbf{s}}_i^s \hat{\mathbf{rs}}_i - Q_i$
end for

return $\begin{bmatrix} \dot{\mathbf{q}}^{k*} - \dot{\mathbf{q}}^k \\ \mathbf{rs} \end{bmatrix}$

1.3.4 Analytic Constraints

The base joint in the multi-body system in Figure 1.2 is typically constrained to some prescribed motion path. In practice, we would want to impose some additional constraints on the system, e.g. constraining a point on some other link to a motion path, or constraining a joint to an animated value in time. These

constraints are transient in nature and often introduce cyclic dependancy in the system. Thus they are treated as external constraints, as opposed to defining them implicitly as part of reduced coordinate formulation.

Initially, we enthusiastically tried the DAE's natural way of defining constraints via algebraic slack variables. The general form of a DAE with algebraic constraints is

$$\mathbf{F}(\mathbf{y}, \dot{\mathbf{y}}, \mathbf{x}, t) = \mathbf{0} \tag{1.10}$$

where $\mathbf{x}$ is the set of algebraic variables. For each constraint, we formulated a scalar valued constraint function $\phi_i(\mathbf{y}, \dot{\mathbf{y}}, t)$ and inserted an algebraic variable associated with the residue corresponding to the constraint condition $x \equiv \phi_i(\mathbf{y}, \dot{\mathbf{y}}, t) = 0$ into the DAE. However, we soon abandoned this line of thinking for the following reasons

- The constraints are transient in nature. We either have to adjust the dimension of algebraic variables $\mathbf{x}$ according to the number of active constraints, or represent all the possible constraints and activate or deactivate them algebraically.
- We found the DAE solver's convergence rate deteriorates rapidly with each additional algebraic constraint. Further, if the constraint can not be satisfied, the integrator does not converge.
- The algebraic constraints can only represent bilateral constraints. The constraints arising from collisions are unilateral. We would have to extend our scope to Differential Variational Inequalities [PS03], which are extension of DAE that handle inequality constraints on differential and algebraic variables.

Instead, we augment the DAE based multi-body formulation inspired by methodologies proposed by [Mir96] and recently by [Kok04] on impulse dynamics and analytical constraints. We would like to give a brief overview of the methodology, along with the details on how we integrate it with the DAE framework and some interesting implementation issues.

Consider a point constraint $\mathbf{p}_j$ as depicted in Figure 1.3. The trajectory of $\mathbf{p}_j$, starting with the initial conditions, can be uniquely defined by the time varying acceleration $a_j^d \mathbf{n}_j$. As discussed in the previous section, we do not directly evaluate the state derivative vectors $\dot{\mathbf{y}}_{t+1}$ in order to integrate the system $\mathbf{y}_t \rightarrow \mathbf{y}_{t+1}$.

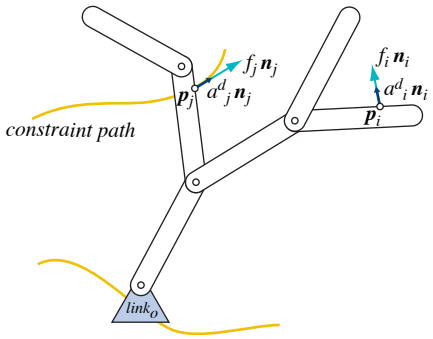

Figure 1.3: Constraints

Therefore, we can not simply enforce the acceleration constraint, by directly altering the state derivatives $\dot{\mathbf{y}}_{t+1}$ as proposed by Kokkevis. We enforce the constraint by applying an external force instead. However, we don't use a penalty like force formulation. Before every DAE integration step, we analytically determine the precise nature of the force $f_j\mathbf{n}_j$, using the similar methodology as in [Kok04]. The unit constraint direction $\mathbf{n}_j$ is treated as constant and is updated for every integration step. There is a linear relationship between the magnitude of the applied force f_j and the resulting desired acceleration a_j^d

$$a_j^d = \frac{\partial a_j}{\partial f_j} f_j + a_j^0 \tag{1.11}$$

where, a_j^0 is the acceleration in the direction $\mathbf{n}_j$ before the force is applied. If we have another constraint point $\mathbf{p}_i$ with force having magnitude f_i in the direction $\mathbf{n}_i$, the resulting accelerations a_i^d and a_j^d will be given by the following linear system

$$\begin{bmatrix} a_i^d \\ a_j^d \end{bmatrix} = \begin{bmatrix} \partial a_i/\partial f_i & \partial a_i/\partial f_j \\ \partial a_j/\partial f_i & \partial a_j/\partial f_j \end{bmatrix} \begin{bmatrix} f_i \\ f_j \end{bmatrix} + \begin{bmatrix} a_i^0 \\ a_j^0 \end{bmatrix} \tag{1.12}$$

Generalizing, for m such constraints we need to determine the vector of $\mathbf{f} \in \Re^m$ unknown force magnitudes by solving the following linear system.

$$\underbrace{\mathbf{K}\mathbf{f} + \mathbf{a}^0}_{\mathbf{a}} - \mathbf{a}^d = \mathbf{0} \tag{1.13}$$

The Jacobian $\mathbf{K} \in \Re^{m \times m}$ can be evaluated by applying unit test force at each constraint and evaluating the changes in accelerations at every constraint. An efficient

procedure to evaluate the Jacobian using the framework of Featherstone's Articulated Body Method is given in [Kok04]. The constraint forces thus determined are applied to the multi-body system over the next integration step via the external force term $\hat{\mathbf{f}}$, as discussed in the previous section.

We replace the constraint direction $\mathbf{n}$ by a spatial vector $\hat{\mathbf{n}}$ to generalize the type of the constraint that can be represented, including the joint acceleration constraint. We further extend the constraint system to include the unilateral constraints such as collisions, friction and joint limits by posing it as a Linear Complementarity Problem (LCP).

$$\underbrace{\mathbf{K}\mathbf{f}+\mathbf{a}^0}_{\mathbf{a}}-\mathbf{a}^d \geq \mathbf{0} \quad \Leftrightarrow \quad \mathbf{f} \geq \mathbf{0} \tag{1.14}$$

The LCP states that forces $\mathbf{f}$, applied only in positive constraint direction, would strive to make the resulting constraint accelerations $\mathbf{a}$ equal to desired acceleration $\mathbf{a}^d$. However, force f_i will be zero if the resulting constraint acceleration a_i is greater than desired acceleration a_i^d. We will discuss the significance of the LCP formulation when we develop the collision response model in the next section.

At first, the solution of a LCP might appear as a daunting task. However, the iterative LCP methods [CPS92] are surprisingly simple and adequate for our purpose. [Ken04] gives a gentle introduction to the solution methods based on various matrix splitting techniques. Apart from the simplicity of the implementation, the iterative LCP solvers have other advantages as compared to pivoting methods. As we will discuss in the next section, we often need to apply multiple constraints on a single link. In this case, the Jacobian $\mathbf{K}$ will have linearly dependent columns. The iterative methods try to distribute the required forces evenly on the link, when multiple solutions exists in this case. Secondly, the LCP may not have a solution. The LCP problems arising from friction models are often non-convex [Bar92, PT96], particularly for high friction values. Further, the Jacobian can be singular or near singular if the limited DOFs of multi-body system does not allow motion in a constraint directions. In all these cases, we can bailout early in the solution process and still have a finite and a well distributed solution for the forces.

We list an iterative LCP solver in Algorithm 2. Apart from the lines 14 and 12 it is a standard successive-over-relaxation linear system solver. Line 14 ensures the inequality condition. We add ε to the diagonal term in line 12 to make $\mathbf{A}$ positive definite, from potentially positive semi-definite, and guard against potentially zero

Algorithm 2 $sor\_lcp(\mathbf{A}, \mathbf{x}, \mathbf{b}, \omega, \varepsilon, K_{iter})$

Require: $\mathbf{A}$ is symmetric, positive semi-definite

Ensure: $\mathbf{w} \equiv \mathbf{A}\,\mathbf{x} - \mathbf{b} \;\geq\; \mathbf{0},\; \mathbf{x} \geq \mathbf{0},\; \mathbf{x}^T\mathbf{w} = 0$

$\mathbf{x} \leftarrow \mathbf{0}$
$n \leftarrow dim(x)$
for $k = 1$ to K_{iter} **do**
  for $i = 1$ to n **do**
    $\delta \leftarrow 0$
    for $j = 1$ to $i - 1$ **do**
      $\delta = \delta + \mathbf{A}_{i,j}\,\mathbf{x}_j$
    end for
    for $j = i + 1$ to n **do**
      $\delta = \delta + \mathbf{A}_{i,j}\,\mathbf{x}_j$
    end for
    $\delta = (\mathbf{b}_i - \delta)/(\mathbf{A}_{i,i} + \varepsilon)$
    $\mathbf{x}_i = (1 - \omega)\mathbf{x}_i + \omega\delta$
    $\mathbf{x}_i = 0 \quad \text{if} \quad \mathbf{x}_i < 0$
  end for
end for

or near zero diagonal terms in the Jacobian $\mathbf{K}$. Further, instead of any elaborate convergence check, we simply make fixed number of iterations K_{iter}, as we know that the solution may not exist. Using forces for enforcing the constraints has an advantage here. If the forces are indeterminate, they get projected into the multi-body's motion null-space, thus always giving valid configuration, without any "pops" in the simulation. Further, as the forces are determined analytically, as compared to, say, penalty based formulation, they are small for most types of the constraints. Thus they are well within the stability zone of the integrator taking 4-8 time-steps per frame. The only exception to this is velocity impulse constraint, we will discuss this case in detail in the next section. As the constraint may not be satisfied accurately, we augment the constraint acceleration by a proportional-derivative form. To exemplify, for a positional constraint, the constraint desired acceleration and the constraint direction be

$$\begin{aligned} \mathbf{a}_i^d = \ddot{\mathbf{p}}_i^d \; + \; K_p(\mathbf{p}_i^d - \mathbf{p}_i) + \; K_d(\dot{\mathbf{p}}_i^d - \mathbf{v}_i) \\ a_i^d = \| \, \mathbf{a}_i^d \, \|, \quad \mathbf{n}_i = \mathbf{a}_i^d / a_i^d \end{aligned} \tag{1.15}$$

where, $\ddot{\mathbf{p}}_i^d, \dot{\mathbf{p}}_i^d, \mathbf{p}_i^d$ are acceleration, velocity and position of the constraint path, and $\mathbf{p}_i, \mathbf{v}_i$ are the current position and velocity of the constraint.

It is important to remove the effect of the constraint forces applied to multi-body system from the previous integration step, and adjust the initial constraint accelerations $\hat{\mathbf{a}}^0$ accordingly, before we determine the next set of constraint forces. We can use the same procedure that determines the Jacobian $\mathbf{K}$ by method of applying test forces for this.

1.3.5 Collision Response

We use the unilateral position constraints discussed in the previous section to develop collision response model for the multi-body system. Collision Detection is a mature subject in computer graphics. For brevity, we only enlist the requirements from the collision detection system for our purpose. Between the current configuration given by the state vector $\mathbf{y}_t$ and extrapolated configuration using derivative vector $\dot{\mathbf{y}}_t$ and next integration time step h, we find all the points on the multi-body system that would collide with the obstacles. Figure 1.4 shows two such collision positions – point $\mathbf{p}_i$ is already penetrated the obstacle and point $\mathbf{p}_j$ is about to collide. There may be more than one colliding point for a link. Let $\mathbf{n}_i$

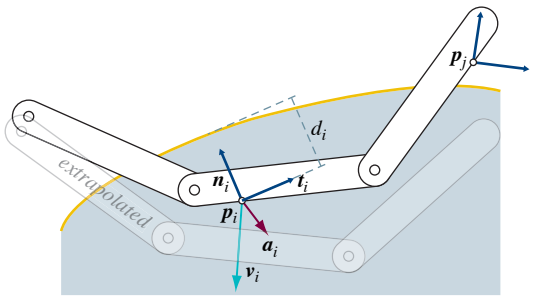

Figure 1.4: Collision as unilateral constraints

be the collision normal, direction directly away from the obstacle, and $\mathbf{a}_i$ and $\mathbf{v}_i$ be collision accelerations and velocities respectively, relative to the obstacle.

We apply collision response in two steps – *contacts* and *impacts*. We first compute the unilateral constraints that would prevent collision points from accelerating towards the obstacle. Followed by computation of velocity impulses that would prevent collision points from moving towards the obstacle.

contacts: We decompose the collision acceleration and the collision velocity into the normal components $\mathbf{a}_{ni}, \mathbf{v}_{ni}$ and tangential components $\mathbf{a}_{ti}, \mathbf{v}_{ti}$. To prevent any acceleration towards the obstacle, we insert a unilateral constraint along the collision normal direction $\mathbf{n}_i$. The unilateral constraint corresponding to the friction acts in the tangent plane defined by the collision normal. We could sample the tangent plane into discrete set of tangents to formulate a complex and numerically expensive friction model based of the discrete frictional cone. Instead, taking inputs from [Kok04], we formulate a novel technique as follows. We set the unilateral constraint direction corresponding to friction as

$$\mathbf{t}_i = unit(\ \mathbf{a}_{ti}\ + \mathbf{v}_{ti}/h\ ) \tag{1.16}$$

If both $\mathbf{a}_{ti}$ and $\mathbf{v}_{ti}$ is zero, we use previously determined tangent vector for the contact. Finally, the LCP formulation corresponding to the two unilateral constraints in the direction $\mathbf{n}_i$ and $\mathbf{t}_i$ at collision position $\mathbf{p}_i$ is

$$\begin{aligned} a_{ni} - a^d_{ni} \geq 0 \quad &\Leftrightarrow \quad f_{ni} \geq 0 \\ \mu_i f_{ni} - f_{ti} \geq 0 \quad &\Leftrightarrow \quad \lambda_i \geq 0 \\ (a_{ti} - a^d_{ti}) + \lambda_i \geq 0 \quad &\Leftrightarrow \quad f_{ti} \geq 0 \end{aligned} \tag{1.17}$$

We set the desired normal acceleration a_{ni}^d proportional to the penetration depth d_i if the point is penetrating, see equation 1.15, or zero if the collision point is outside the obstacle. The desired tangential acceleration a_{ti}^d is set to $(-\parallel \mathbf{v}_{ti} \parallel /h)$. The LCP formulation will compute required amount of normal force f_{ni} to remove the normal acceleration $\mathbf{a}_{ni}$. The tangential force f_{ti} will be at most $\mu_i f_{ni}$, and try to remove any tangential non-zero velocity component – the dynamic friction case, or if the tangential velocity is zero it will try to remove any tangential acceleration – the static friction case.

impacts: We use impulses to arrest the collision normal velocity $\mathbf{v}_{ni}$. Only those contacts are considered that have the normal velocity component $v_{ni} < 0$. For the impulse computations we can use the same acceleration constraints discussed in the previous section by setting $a_{ni}^d = -(1+\nu)v_{ni}$, where ν is collision restitution. Instead of applying potentially large forces, we alter the joint velocities $\dot{\mathbf{q}}_t$. This would invalidate the consistent solution set $\{\mathbf{y}_t, \dot{\mathbf{y}}_t\}$. We should correct $\ddot{\mathbf{q}}$ correspondingly. In reality, we found that the solver is tolerant to the error.

1.3.6 Implementation Details

Having developed the theoretical framework in the last three sections, in this section we would like to give a brief overview of the Oriented Strands system modeled as dynamics of multi-body system. It is implemented as a plug-in to Maya, as well as plug-in to our proprietary animation system. We use DASPK [BCP96] for our implementation.

Along with the robust formulation, any physically based simulation system to be successful in production environment needs to have an important aspect – to be able to art direct. In the following subsections, we develop some novel concepts towards that, along with few important implementation details. In our opinion, choice of reduced coordinate formulation and dynamics expressed in local frames makes some of these concept easier to implement.

1.3.7 Simulation Parameters and Workflow

The dynamic strand is composed from a hierarchy of input curves defined in a local frame, that defines the initial rest shape of the corresponding multi-body

system. We provide the user with high-level control over the direct numerical simulation by means of relevant physical parameters of the dynamic strand, such as *mass per unit length, strand radius, bend stiffness/damping, torsional stiffness/damping, gravity, air drag*. The user can animate all the parameters and specify their length-wise anisotropic variation. The collision parameters *collision restitution and static/dynamic friction* are defined per obstacle surface. The strand may have additional anisotropic weights over collision parameters, along with their length-wise variation.

1.3.8 Stiffness Model and External Forces

In Section 1.3.3, while developing the DAE based formulation, we assumed single-DOF joints for the simplicity of discussion. However, we use three-DOF spherical joint in the implementation of Oriented Strands. The joint variable of i^{th} joint is expressed as a quaternion $\mathbf{q}_i \in \Re^4$ and the joint velocity as a vector $\mathbf{w}_i \in \Re^3$. [Had03, Fea87] gives details on how to formulate multiple-DOF joints.

We decompose the quaternion defining the relative transformation between two links into a twist component θ_t around the local y-axis and a pure bend component θ_b around a bend axis $\mathbf{b}$. We provide a nonlinear stiffness model as follows

$$\begin{aligned} \mathbf{Q}_b &= K_b(\mathbf{b})\ \mathbf{b}\ \tan((\theta_b - \theta_b^0)/2) \\ \mathbf{Q}_t &= K_t\ \mathbf{y}\ \tan((\theta_t - \theta_t^0)/2) \end{aligned} \tag{1.18}$$

where θ_b^0 and θ_t^0 correspond to the rest configuration. K_t is *torsional stiffness* coefficient and $K_b(\mathbf{b})$ is anisotropic *bend stiffness* coefficient. The response model is almost linear for small deformations. However, the non-linear response prevents excessive deformations and potentially joints snapping.

We support general external force fields using the plug-in architecture of Maya and that of our proprietary animation system. The user can attach any complex combination of time-varying fields such as wake, turbulence, fluid simulations and general event driven scripted force fields. The user can further specify length-wise anisotropic weights for the external force fields. The user can optionally include these fields in computing the Jacobians numerically discussed in Section 1.3.3.

1.3.9 Accurate Acceleration of Base Joint

In the reduced coordinate formulation it is critical to compute and supply the accurate velocities and accelerations of the base joint's prescribed motion path. We could have evaluated them numerically, however that would mean making repetitive evaluations of motion system at sub-frame interval, which is typically very expensive. Instead we interpolate the rigid-body transformation from four successive frames. Constructing a C_2 continuous curve that interpolates a number of prescribed rotations is a well studied problem. We use the method developed by [PK96], where we construct a piecewise cubic curve whose coefficients a_i, b_i, c_i are members of $so(3)$. The rotation is evaluated by taking the matrix exponential of this polynomial.

1.3.10 Time Scale and Local Dynamics

Often the dynamical simulation are encountered with very extreme and brisk animated motions. Although a robust formulations will be able to cope with the scenario, often the directors would want the motion to be selectively less violent. We introduce *time scale* β to control the amount of energy pumped in the system. It is a factor by which velocity and acceleration of the base joint get scaled and is typically between zero and one, however the user can set it more than one to accentuate the motion. The *local dynamics* γ is another similar parameter which blends out velocity and acceleration of some local dynamics reference frame.

$$\begin{aligned} \hat{\mathbf{a}}_0 &= \beta\ (\hat{\mathbf{a}}_0 - \gamma\, \hat{\mathbf{a}}_{ref}) \\ \hat{\mathbf{v}}_0 &= \beta\ (\hat{\mathbf{v}}_0 - \gamma\, \hat{\mathbf{v}}_{ref}) \end{aligned} \tag{1.19}$$

One scenario that is frequent is, a braid or long hair that fly away when character starts running or rides a horse. The local dynamics reference frame is simply set to the character's hip joint, and with appropriate *local dynamics* parameter one can control the amount of flyaway the user wants.

1.3.11 Posable Dynamics

Ears and tail, often have free secondary dynamic motion when the animator lets them "loose". However, animator would want to hit a certain pose at a certain time

to make the character expressive. We tried different techniques that are based on the motion control principle. However, it did not give desired results. For high values of K_p and K_d in the PID controller (Equation 1.15), the constraint follows the goal rather exactly. If we reduce K_p and K_d, due to *slew rate*, the PID controller gave a large error in achieving pose and the solution oscillated a lot before coming to rest to the animated pose. Surprisingly a very simple model worked for this specific application. We insert a spring between the dynamic strand and the desired animated pose at tip of each segment, to give a penalty based "soft" constraint. The user can animate the stiffness and damping, namely *pose strength* and *pose damping* to achieve the Posable Dynamics.

1.3.12 Zero-gravity Rest Shape

The rest shapes of the dynamic strands are typically modeled taking the gravity into account. Intricate hairstyle is a good example of this. Unfortunately, when we start simulating hair, the strands would sag under the gravity before they finally settles down. This would change the original art directed hair shape depending on the stiffness. Increasing the stiffness to preserve the shape would give unrealistic motion. One can go back and try to edit the rest shape so as to achieve desired shape under gravity. However, this would be very laborious and iterative process. We developed a technique to automatically compute the equivalent rest shape, without gravity, so that under gravity we would get the desired shape. The problem is a straight forward *inverse dynamics* problem in robotics. Given a set of external forces (gravity) and given the desired configuration of multi-body system, *inverse dynamics* problem finds set of joint forces required to achieve certain joint accelerations, zero in our case. We refer to [Fea87] for the details. We would like to highlight that it would be difficult to formulate this in the case of maximal coordinate formulation.

1.3.13 Strand-strand Interaction

Strand-strand interaction is not important in some simulation scenarios such as foliage motion, braids and ears, whereas it is critical in certain cases such hair simulation. We have implemented a modular plug-in field to Maya that computes the fluid like forces on a continuum, that can be attached to the Oriented Strands system to realize the strand-strand interactions as introduced by [HMT01b]. The

other interesting approaches to handle strand-strand interactions include wisp level interactions [PCP01b, BKCN03b], layers [LK01b] and strips [CJY02b].

We demonstrate the effectiveness of the proposed Oriented Strand methodology, through impressive results in production of Madagascar and Shrek The Third at PDI/DreamWorks, in Section 5.1.

1.4 Super-Helices: a compact model for thin geometry

Figure 1.5: Left, a Super-Helix. Middle and right, dynamic simulation of natural hair of various types: wavy, curly, straight. These hairstyles were animated using $N = 5$ helical elements per guide strand.

The Super-Helix model is a novel mechanical model for hair, dedicated to the accurate simulation of hair dynamics. In the spirit of work by Marschner *et al.* in the field of hair rendering [MJC+03a], we rely on the structural and mechanical features of real hair to achieve realism. This leads us to use Kirchhoff equations for dynamic rods. These equations are integrated in time thanks to a new deformable model that we call *Super-Helices*: A hair strand is modeled as a C^1 continuous, piecewise helical[1] rod, with an oval to circular cross section. We use the degrees of freedom of this inextensible rod model as generalized coordinates, and derive the equations of motion by Lagrangian mechanics. As our validations show, the resulting model accurately captures the nonlinear behavior of hair in motion, while ensuring both efficiency and robustness of the simulation.

This work was published at SIGGRAPH in 2006 [BAC+06], and results from a collaboration with Basile Audoly, researcher in mechanics at Universite Pierre et Marie Curie, Paris 6, France.

[1] A helix is a curve with constant curvatures and twist. Note that this definition includes straight lines (zero curvatures and twist), so Super-Helices can be used for representing any kind of hair.

1.4.1 The Dynamics of Super-Helices

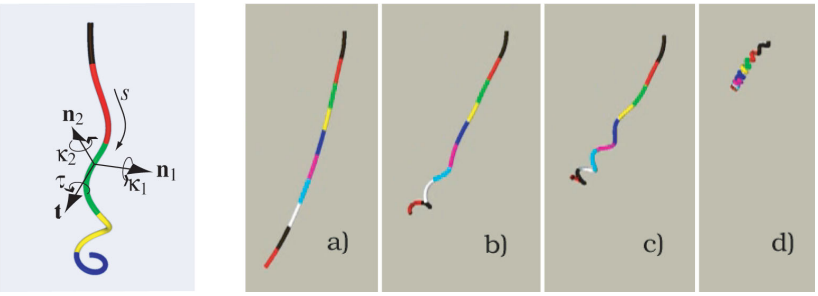

Figure 1.6: Left, geometry of Super-Helix. Right, animating Super-Helices with different natural curvatures and twist: a) straight, b) wavy, c) curly, d) strongly curly. In this example, each Super-Helix is composed of 10 helical elements.

We shall first present the model that we used to animate individual hair strands (guide strands). This model has a tunable number of degrees of freedom. It is built upon the Cosserat and Kirchhoff theories of rods. In mechanical engineering literature, a rod is defined as an elastic material that is effectively one dimensional: its length is much larger than the size of its cross section.

Kinematics

We consider an inextensible rod of length L. Let $s \in [0,L]$ be the curvilinear abscissa along the rod. The *centerline*, $\mathbf{r}(s,t)$, is the curve passing through the center of mass of every cross section. This curve describes the shape of the rod at a particular time t but it does not tell how much the rod *twists* around its centerline. In order to keep track of twist, the *Cosserat model* introduces a material frame $\mathbf{n}_i(s,t)$ at every point of the centerline[2]. By *material*, we mean that the frame 'flows' along with the surrounding material upon deformation. By convention, $\mathbf{n}_0$ is the tangent to the centerline:

$$\mathbf{r}'(s,t) = \mathbf{n}_0(s,t), \tag{1.20a}$$

[2]By convention, lowercase Latin indices such as i are used for all spatial directions and run over $i = 0,1,2$ while Greek indices such as α are for spatial directions restricted to the plane of the cross section, $\alpha = 1,2$.

while $(\mathbf{n}_\alpha)_{\alpha=1,2}$ span the plane of the cross section, see Figure 1.6, left. We use primes to denote space derivatives along the center line, $f' = \partial f/\partial s$, while the overstruck notation is for time derivatives, $\dot{f} = \mathrm{d}f/\mathrm{d}t$.

The Kirchhoff model for elastic rod starts from this mathematical description of a Cosserat curve and adds the physical requirement of inextensibility and unshearability. In this case, the frame $(\mathbf{n}_i(s))_{i=0,1,2}$ is orthonormal for all s, and there exists a vector $\mathbf{\Omega}(s,t)$, called the Darboux vector, such that:

$$\mathbf{n}_i'(s,t) = \mathbf{\Omega}(s,t) \times \mathbf{n}_i(s,t) \quad \text{for } i = 0,1,2. \tag{1.20b}$$

Appropriate boundary conditions must be specified: one end of the hair strand, $s = 0$, is clamped into the head while the other end, $s = L$, is free. The position of the clamped end, together with the orientation of the initial frame, are imposed by head motion (an input in the simulations):

$$\begin{cases} \mathbf{r}(0,t) = \mathbf{r}_\mathrm{c}(t) \\ \mathbf{n}_i(0,t) = \mathbf{n}_{i,\mathrm{c}}(t) \quad \text{for } i = 0,1,2, \end{cases} \tag{1.20c}$$

where subscript 'c' refers to the *clamped* end of the rod, $s = 0$.

The rod's material curvatures $(\kappa_\alpha(s,t))_{\alpha=1,2}$ with respect to the two directions of the cross section and the twist $\tau(s,t)$ are defined as the coordinates of the vector $\mathbf{\Omega}(s,t)$ in the local material frame:

$$\mathbf{\Omega}(s,t) = \tau(s,t)\,\mathbf{n}_0(s,t) + \kappa_1(s,t)\,\mathbf{n}_1(s,t) + \kappa_2(s,t)\,\mathbf{n}_2(s,t). \tag{1.21}$$

By introducing a redundant notation for the twist, $\kappa_0 = \tau$, we can refer to these parameters collectively as $(\kappa_i(s,t))_{i=0,1,2}$.

Reconstruction, generalized coordinates

The degrees of freedom of a Kirchhoff rod are its material curvatures and twist $(\kappa_i(s,t))_{i=0,1,2}$. A continuous model being of little use for computer animation, we introduce a spatial discretization as follows. Let us divide the strand $s \in [O,L]$ into N segments S_Q indexed by Q ($1 \le Q \le N$). These segments may have different lengths, and N is an arbitrary integer, $N \ge 1$. We define the material curvatures and twist of our deformable model with piecewise constant functions over these segments. We write $q_{i,Q}(t)$ the constant value of the curvature κ_i (for $i = 1,2$)

or twist $\kappa_0 = \tau$ (for $i = 0$) over the segment S_Q at time t. Therefore, an explicit formula for the material curvatures and twist reads

$$\kappa_i(s,t) = \sum_{Q=1}^{N} q_{i,Q}(t)\, \chi_Q(s) \tag{1.22}$$

where $\chi_Q(s)$ is the characteristic function of segment Q, equal to 1 if $s \in S_Q$ and 0 otherwise. We collect the numbers $q_{i,Q}(t)$ into a vector $\mathbf{q}(t)$ of size $3N$, which we call the *generalized coordinates* of our model.

These generalized coordinates $\mathbf{q}(t)$ can be used to reconstruct the rod shape at any given time. Indeed, plugging equation (1.22) into equation (1.21), and then equation (1.21) into equations (1.20a–c) yields a differential equation with respect to s. By integration of this equation, one obtains the centerline $\mathbf{r}(s)$ and the material frames $\mathbf{n}_i(s)$ as a function of s and $\mathbf{q}(t)$. This process, called the *reconstruction*, can be carried out analytically; as explained in Appendix 1.4.3, the integration with respect to s has a symbolic solution over every segment S_Q. By patching these solutions, we find that our model deforms as a helix over every segment S_Q and, moreover, is C^1-smooth (between adjacent helices, both the centerline and the material frames are continuous). This is why we call this model a *Super-Helix*. We write $\mathbf{r}^{\mathrm{SH}}(s,\mathbf{q})$ and $\mathbf{n}_i^{\mathrm{SH}}(s,\mathbf{q})$ as the parameterization of the Super-Helix in terms of its generalized coordinates $\mathbf{q}$. In Appendix 1.4.3, we explain how these functions $\mathbf{r}^{\mathrm{SH}}$ and $\mathbf{n}_i^{\mathrm{SH}}$ can be obtained in symbolic form.

Imposing a uniform value to the material curvatures and twist over the hair length would make it deform as a plain helix. This is indeed what happens when one chooses the coarsest possible spatial discretization, that is $N = 1$. For other values of N, the rod is made of several helices patched together. Large values of N yield arbitrarily fine space discretizations.

Dynamic equations for a Super-Helix

Given a deformable body whose configuration depends on generalized coordinates $\mathbf{q}(t)$, Lagrangian mechanics provides a systematic method for deriving its equations of motion, $\ddot{\mathbf{q}} = \mathbf{a}(\mathbf{q},\dot{\mathbf{q}},t)$. This is done by feeding the Lagrangian equations of motion:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_{iQ}}\right) - \frac{\partial T}{\partial q_{iQ}} + \frac{\partial U}{\partial q_{iQ}} + \frac{\partial D}{\partial \dot{q}_{iQ}} = \int_0^L \mathbf{J}_{iQ}(s,\mathbf{q},t)\cdot\mathbf{F}(s,t)\,\mathrm{d}s \tag{1.23}$$

with the expressions for the kinetic energy $T(\mathbf{q},\dot{\mathbf{q}},t)$, for the internal energy $U(\mathbf{q},t)$ and for the dissipation potential $D(\mathbf{q},\dot{\mathbf{q}},t)$ that describe the physics of the system at hand. The right-hand side of equation (1.23) is the generalized force f_{iQ} deriving from the lineic density $\mathbf{F}(s,t)$ of physical force applied on the rod, and $\mathbf{J}_{iQ}$ defines the Jacobian matrix, $\mathbf{J}_{iQ} = \partial \mathbf{r}^{\mathrm{SH}}(s,\mathbf{q})/\partial q_{iQ}$. We consider three force contributions, namely hair weight, viscous drag from ambient air (considered at rest for simplicity) with coefficient ν, and interaction forces with surrounding strands and body:

$$\mathbf{F}(s,t) = \rho S \mathbf{g} - \nu \dot{\mathbf{r}}^{\mathrm{SH}}(s,\mathbf{q}) + \mathbf{F}^{\mathrm{i}}(s,t), \tag{1.24a}$$

where $\mathbf{F}(s,t)$ is the total external force applied to the rod per unit length, ρS is the mass of the rod per unit length, and $\mathbf{g}$ is the acceleration of gravity. The interaction forces $\mathbf{F}^{\mathrm{i}}$ are computed using the model presented shortly in Section 1.4.2.

The three energies in the equations of motion (1.23) that are relevant for an elastic rod are:

$$T(\mathbf{q},\dot{\mathbf{q}},t) = \frac{1}{2}\int_0^L \rho S \left(\dot{\mathbf{r}}^{\mathrm{SH}}(s,\mathbf{q})\right)^2 \mathrm{d}s \tag{1.24b}$$

$$U(\mathbf{q},t) = \frac{1}{2}\int_0^L \sum_{i=0}^{2} (EI)_i \left(\kappa_i^{\mathrm{SH}}(s,\mathbf{q}) - \kappa_i^{\mathrm{n}}(s)\right)^2 \mathrm{d}s \tag{1.24c}$$

$$D(\mathbf{q},\dot{\mathbf{q}},t) = \frac{1}{2}\int_0^L \gamma \sum_{i=0}^{2} \left(\dot{\kappa}_i^{\mathrm{SH}}(s,\mathbf{q})\right)^2 \mathrm{d}s. \tag{1.24d}$$

The kinetic energy T is defined in terms of the rod velocity, $\dot{\mathbf{r}} = \mathrm{d}\mathbf{r}/\mathrm{d}t$ in the classical way. The internal energy U in equation (1.24c) is the elastic energy of a rod, as derived, for instance, in [AP07] and used in [BAQ+05]. The coefficients $(EI)_i$ are the principal bending stiffness of the rod in the directions $\mathbf{n}_i$ (for $i = 1,2$) while $(EI)_0$ is the torsional stiffness, classically written μJ (for $i = 0$). These parameters are given by textbook formulas in terms of the material properties (Young's modulus and Poisson's ratio) and of the geometry of the cross-section. The quantities $\kappa_i^{\mathrm{n}}(s)$ are called the *natural curvatures ($i = 1,2$) and twist ($i = 0$)* of the rod. They characterize the shape of the rod in the absence of external force: for $\kappa_i(s) = \kappa_i^{\mathrm{n}}(s)$ the elastic energy is vanishing and therefore minimum. Vanishing natural curvatures ($\kappa_\alpha^{\mathrm{n}} = 0$ for $\alpha = 1,2$) model straight hair. Nonzero values will result in wavy, curly or fuzzy hair. In practice, tuning these parameters allows one to choose for the desired hair style, as explained in Section **??**. Overall, the mechanical properties of the rod are captured by only six entities, the stiffnesses $(EI_i)_{i=0,1,2}$ and the natural twist and curvatures $(\kappa_i^{\mathrm{n}}(s))_{i=0,1,2}$. We neglect the dependence of the stiffnesses on s, but not that of the natural twist and curvatures:

we found that slight variations of $(\kappa_i^{\mathrm{n}}(s))_i$ with s allow for more realistic hair styles. Finally, we choose for the dissipation energy D in equation (1.24d) a simple heuristic model for capturing visco-elastic effects in hair strands, the coefficient γ being the internal friction coefficient.

All the terms needed in equation (1.23) have been given in equations (1.24). By plugging the latter into the former, one arrives at explicit equations of motion for the generalized coordinate $\mathbf{q}(t)$. Although straightforward in principle, this calculation is involved[3]. It can nevertheless be worked out easily using a symbolic calculation language such as Mathematica [Wol99]: the first step is to implement the reconstruction of Super-Helices as given in Appendix 1.4.3; the second step is to work out the right-hand sides of equations (1.24), using symbolic integration whenever necessary; the final step is to plug everything back into equation (1.23). This leads to the equation of motion of a Super-Helix:

$$\mathbb{M}[s,\mathbf{q}]\cdot\ddot{\mathbf{q}}+\mathbb{K}\cdot(\mathbf{q}-\mathbf{q}^{\mathrm{n}})=\mathbf{A}[t,\mathbf{q},\dot{\mathbf{q}}]+\int_0^L\mathbf{J}_{iQ}[s,\mathbf{q},t]\cdot\mathbf{F}^{\mathrm{i}}(s,t)\,\mathrm{d}s. \tag{1.25}$$

In this equation, *the bracket notation is used to emphasize that all functions are given by explicit formula in terms of their arguments*.

In equation (1.25), the inertia matrix $\mathbb{M}$ is a dense square matrix of size $3N$, which depends nonlinearly on $\mathbf{q}$. The stiffness matrix $\mathbb{K}$ has the same size, is diagonal, and is filled with the bending and torsional stiffnesses of the rod. The vector $\mathbf{q}^{\mathrm{n}}$ defines the rest position in generalized coordinates, and is filled with the natural twist or curvature κ_i^{n} of the rod over element labelled Q. Finally, the vector $\mathbf{A}$ collects all remaining terms, including air drag and visco-elastic dissipation, which are independent of $\ddot{\mathbf{q}}$ and may depend nonlinearly on $\mathbf{q}$ and $\dot{\mathbf{q}}$.

Time discretization

The equation of motion (1.25) is discrete in space but continuous in time. For its time integration, we used a classical Newton semi-implicit scheme with fixed time step. Both the terms $\ddot{\mathbf{q}}$ and $\mathbf{q}$ in the left-hand side are implicited. Every time step involves the solution of a linear system of size $3N$. The matrix of this linear system is square and dense, like $\mathbb{M}$, and is different at every time step: a

[3]The elements of $\mathbb{M}$, for instance, read $M_{iQ,i'Q'}=\frac{1}{2}\iint\mathbf{J}_{iQ}(s,\mathbf{q})\cdot\mathbf{J}_{i'Q'}(s',\mathbf{q})\,\mathrm{d}s\,\mathrm{d}s'$ where $\mathbf{J}$ is the gradient of $\mathbf{r}^{\mathrm{SH}}(s,\mathbf{q})$ with respect to $\mathbf{q}$.

conjugate-gradient algorithm is used. The density of $\mathbb{M}$ is the price to be paid for incorporating the inextensibility constraint into the parameterization. It results in degrees of freedom that are non local in physical space.

Super-Helices for solving the Kirchhoff equations

The equations of motion for dynamic elastic rods were derived by Kirchhoff in 1859. A modern derivation of these equations can be found, for instance, in [AP07]: it follows the same principles as the one for a Super-Helix. The main difference is that we have constrained the material curvatures and twists to be piecewise constant functions of s in equation (1.22); these functions may depend arbitrarily on s for regular Kirchhoff rods. Apart from this difference, the Super-Helix model is based on the same physical assumptions as the Kirchhoff equations. Therefore, the Super-Helix method provides a *discrete model for solving the Kirchhoff equations*.

We derived the Super-Helix model after we extensively tested existing integration schemes for the Kirchhoff equations, and eventually realized that they were not well suited for computer graphics applications. We implemented an elegant algorithm, due to [HKS98], based on a discretization of these equations by finite differences. In this paper, Hou *et al.* discuss very clearly the difficulties associated with the numerical integration of the Kirchhoff equations, which are numerically very stiff. They propose an attempt for removing this stiffness. It brings a very significant improvement over previous methods but we found that it was still insufficient for hair animation purposes: there remain quite strong constraints on the time steps compatible with numerical stability of the algorithm. For instance, simulation of a 10 cm long naturally straight hair strand using the algorithm given in [HKS98] remained unstable even with 200 nodes and a time step as low as 10^{-5} s. The stiffness problems in nodal methods have been analyzed in depth by [BW92] who promoted the use of *Lagrangian deformable models* (sometimes called 'global models' as opposed to nodal ones). This is indeed the approach we used above to derive the Super-Helix model, in the same spirit as [WW90, BW92, QT96].

We list a few key features of the Super-Helix model which contribute to realistic, stable and efficient hair simulations. All space integrations in the equations of motion are performed symbolically off-line, leading to a quick and accurate evaluation of the coefficients in the equation of motion at every time step. The

inextensibility constraint, enforced by equations (1.20a–1.20b), is incorporated into the reconstruction process. As a result, the generalized coordinates are free of any constraint and *the stiff constraint of inextensibility has been effectively removed* from the equations. Moreover, the method offers a well-controlled space discretization based on Lagrangian mechanics, leading to stable simulations even for small N. For $N \to \infty$, the Kirchhoff equations are recovered, making the simulations very accurate. By tuning the parameter N, one can freely choose the best compromise between accuracy and efficiency, depending on the complexity of hair motion and on the allowed computational time. We are aware of another Lagrangian model[4] used in computer graphics that provides an adjustable number of degrees of freedom, namely the Dynamic NURBS model [QT96], studied in the 1D case by [NR01]. Finally, external forces can have an arbitrary spatial dependence and do not have to be applied at specific points such as nodes, thereby facilitating the combination with the interaction model.

1.4.2 Applications and Validation

In this section, we provide a validation of our physical model against a series of experiments on real hair, and demonstrate that the Super-Helix model accurately simulates the motion of hair. Images and videos showing our set of results are available at http://www-evasion.imag.fr/Publications/2006/BACQLL06/.

Choosing the parameters of the model

In our model, each Super-Helix stands for an individual hair strand placed into a set of neighboring hair strands, called *hair clump*, which is assumed to deform continuously. To simulate the motion of a given sample of hair, which can either be a hair wisp or a full head of hair, we first deduce the physical and geometric parameters of each Super-Helix from the structural and physical properties of the hair strands composing the clump. Then, we adjust friction parameters of the model according to the damping observed in real motion of the clump. Finally, interactions are set up between the Super-Helices to account for contacts occurring

[4] In this model, *geometric parameters*, defined by the NURBS control points and the associated weights, are used as generalized coordinates in the Lagrangian formalism. In contrast, we opt here for *mechanically-based* generalized coordinates: they are the values of the material curvatures and twist, which are the canonical unknowns of the Kirchhoff equations.

between the different animated hair groups. In this section, we explain how we set all the parameters of the Super-Helix model using simple experiments performed on real hair.

Hair mass and stiffness: We set the density ρ to be equal to a typical value for hair, $1.3\ \mathrm{g}\cdot\mathrm{cm}^{-3}$. The mean radius r and the ellipticity $e = \frac{r_{max}}{r_{min}}$ of the Super-Helix cross-section are deduced by direct microscopic observation of real hair fibers (see Figure 1.7, left) whereas Young's modulus and Poisson's ratio are taken from existing tables, which report values for various ethnic origins [Rob02]. These parameters are then used to compute the bending and torsional stiffnesses $(EI)_{i=0,1,2}$ of the Super-Helix, as given by textbook formulas.

Natural curliness: The natural curvatures and twist parameters of the Super-Helix model are set by:

$$\kappa_1^{\mathrm{n}} = 1/r_{\mathrm{h}} \qquad \kappa_2^{\mathrm{n}} = 0 \qquad \tau^{\mathrm{n}} = \frac{\Delta_{\mathrm{h}}}{2\pi r_{\mathrm{h}}^2},$$

where r_{h} is the radius and Δ_{h} the step of the approximate helical shape of the real hair clump, measured near the tips (see Figure 1.7, right). Indeed, the actual curvatures and twist should be equal to their natural value at the free end of the rod, where the role of gravity becomes negligible. In practice, we add small random variations to these values along each Super-Helix to get more natural results. We have noted that in reality, most hair types have an almost zero natural twist τ^{n}, except African hair (see Appendix 1.4.4).

Internal friction γ: This parameter measures the amount of internal dissipation within a Super-Helix during motion. It especially accounts for the hair-hair dissipative interactions occurring inside the hair clump whose motion is guided by the Super-Helix. We found that, in practice, the internal friction can be easily adjusted by comparing the amplitude of deformation between the real and the simulated hair clump when a vertical oscillatory motion is imposed, see Figure 1.8. Typically, we obtained best results with $\gamma \in [5.10^{-10}, 5.10^{-11}]\ \mathrm{kg}\cdot\mathrm{m}^3\cdot\mathrm{s}^{-1}$.

Air-hair friction coefficient: Once parameter γ is chosen, the air-hair friction parameter can be fitted by comparing the damping duration between the real and

Figure 1.7: Left, measuring the mean radius r and the ellipticity e of the model by observation of real hair fibers with a video-microscope. Right, measuring the radius r_h and the step Δ_h of the natural helical shape at the tip of a real hair clump.

the simulated hair clump, for example when imposing a pendulum motion. We noted the air-hair friction parameter is strongly related to the local alignment of neighboring hair strands, called the *hair discipline* in the field of cosmetics. As one can observe in the real world, fuzzy hair is more subject to air damping than regular, disciplined hair. In practice, we chose the air-hair friction coefficient ν between 5.10^{-6} $\mathrm{kg}\cdot(\mathrm{m}\cdot\mathrm{s})^{-1}$ (disciplined hair) and 5.10^{-5} $\mathrm{kg}\cdot(\mathrm{m}\cdot\mathrm{s})^{-1}$ (fuzzy hair).

Friction with another object: Contacts between hairs, and between our hair model and external objects (such as the body) are performed through penalty forces which include a normal elastic response together with a tangential viscous friction force. For simulating realistic contacts between hair and external objects, we use an anisotropic friction force, which accounts for the oriented scales covering individual hair fibers. The friction parameter is directly adjusted from real observations of sliding contacts between the hair clump and a given material, and then multiplied by a cosine function to account for the orientation of hair fibers with respect to their sliding motion over the external object.

Visual comparisons

With simulation we have reproduced a series of real experiments on smooth and wavy hair clumps to show that our model captures the main dynamic features of

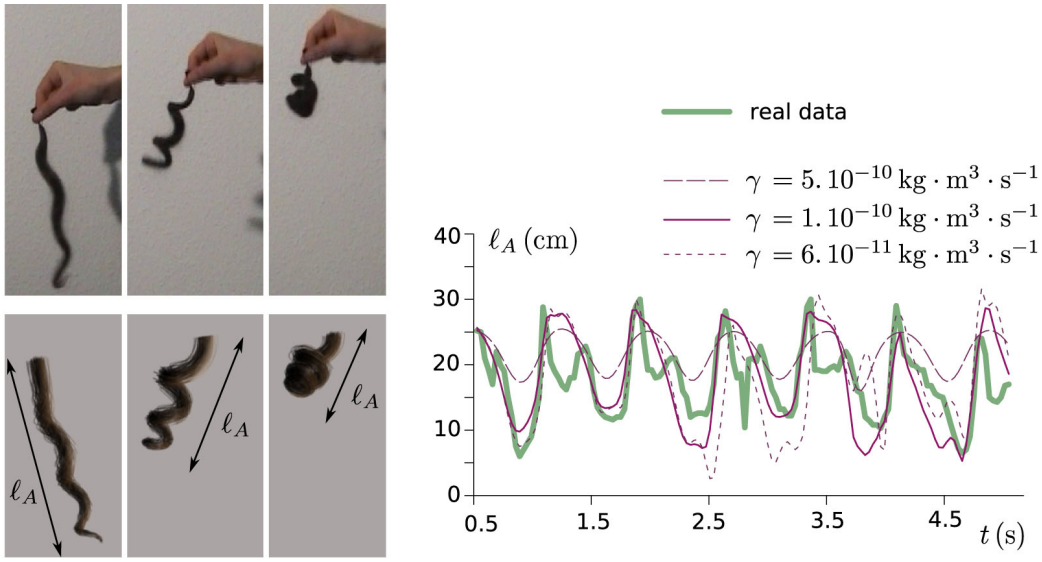

Figure 1.8: Fitting γ for a vertical oscillatory motion of a disciplined, curly hair clump. Left, comparison between the real (top) and virtual (bottom) experiments. Right, the span ℓ_A of the hair clump for real data is compared to the simulations for different values of γ. In this case, $\gamma = 1.10^{-10}\ \mathrm{kg}\cdot\mathrm{m}^3\cdot\mathrm{s}^{-1}$ gives qualitatively similar results.

natural hair. We used the technique presented previously to fit the parameters of the Super Helix from the real manipulated hair clump. As shown in Figure 1.9, left, our Super-Helix model adequately captures the typical nonlinear behavior of hair (buckling, bending-twisting instabilities), as well as the nervousness of curly hair when submitted to high speed motion (see Figure 1.8, left). Figure 1.9, right, shows the fast motion of a large hair, which is realistically simulated using 3 interacting Super-Helices. All these experiments also allowed us to check the stability of the simulation, even for high speed motion.

Finally, Figure 1.10 demonstrates that our model convincingly captures the complex effects occurring in a full head of hair submitted to a high speed shaking motion.

Figure 1.9: Left, buckling effect caused by vertical oscillations of a hair clump. Right, a more complex hair wisp animated with 3 interacting Super-Helices, during fast motion.

Results and simulation performance

Figure 1.5 shows three examples of motion for a full head of hair. Different hair types were simulated, from long to short and curly to straight. To set up our simulations, we used typical parameter values for real hair of different ethnic origins. These parameters are given in Appendix 1.4.4. We used one hundred guide strands for the wavy and curly hairstyles, and two hundred for the smooth Asian hairstyle.

For all hair types, even long or curly ones, we found it to be unnecessary to use more than 5 to 10 helical elements per guide hair strand. For higher values of N, the increase in accuracy becomes imperceptible.

Our model was tested on a 3 GHz Pentium 4 processor. Up to 10 strands can be simulated in real-time. When simulating a full head of hair, we obtained a very reasonable mean computational time of 0.3 s to 3 s per frame. The performance of our implementation is thus as good as other recent approaches, such as [CCK05a]. This is due to the stability of the Super-Helix model, which allows time steps of $\approx 1/30$ s, even during high speed motion, and to the high order of interpolation provided by the helices, which helps to keep N small while offering a good accuracy.

Figure 1.10: Comparison between a real full head of hair and our model, on a head shaking motion (straight and clumpy hair type).

Limitations and future work

The Super-Helix model remains stable for any number N of helical elements in guide strands. However, the matrix $\mathbb{M}$ used in the dynamic computation is dense, and as a result, the computation time increases quickly with N, as $\mathcal{O}(N^2)$. This quadratic time complexity prevents the use of Super-Helices for a very fine simulation. However, this proves to be a minor concern for hair animation purposes, as we find N does not have to be very large for generating pleasant visual results. Moreover, once the number of helical parts is chosen, the complexity of the whole simulation remains linear with respect to the number of guide strands.

Besides this, constraints are currently treated using penalty methods. Analytical methods would be very useful, as they would allow solid friction to be handled. This is one of the planned future extensions of the model.

Although we could advance in the understanding on collective hair behavior, not enough data were available for us to set up the really strong model we would have dreamed of. Indeed, processing non-simulated hair strands by a simple interpolating scheme between a fixed set of sparse guide hair strands may lose fine-scale

details; moreover, when thin objects interact with such sparse hair strands, the coarse granularity of hair may become obvious and distracting. Quantifying the tendency of hair to cluster and separate according to the hair type as well as to the collisions occurring between hair and external objects would be a very interesting avenue for future work. The relationship between this and the intuitive notions of curliness and discipline could be investigated.

1.4.3 Conclusion

We have introduced a deformable model able to simulate hair dynamics for a wide range of hair types, capturing the complex motions observed in real hair motions. In particular, the simulation of curly hair, a notoriously difficult problem, has been demonstrated. *Super-Helices* are based on Kirchhoff equations for elastic, inextensible rods and on Lagrangian dynamics, and provide a freely adjustable number of degrees of freedom. They take into account important hair features such as the natural curvature and twist of hair strands, as well as the oval shape of their cross section. To stress on the powerful representation of moving hair by Super-Helices, we have presented a rigorous validation of this model, supported by a series of comparative experiments on real hair. We also noted that Super-Helices are able to achieve realistic motions at a very reasonable computational cost: this is permitted by the stability of the method, which enables large time steps, and by the high order of interpolation provided by the helices.

An interesting direction for future research would be to adapt our hair model to a real-time framework, in order to perform interactive hair-styling operations or to use it for character animation in video-games. We could think of setting up an adaptive version of the Super-Helices model, where the number of helical parts would automatically vary over time according to the current deformation and to the available computational power, following work in articulated body dynamics [RGL05a].

Appendix

Helical solution

We show here that the reconstruction of the rod can be carried out over any particular element $S_Q = [s_Q^{\mathrm{L}}, s_Q^{\mathrm{R}}]$ of the Super-Helix, over which the functions $(\kappa_i(s))_i$ are constant by construction. By equations (1.20), $\boldsymbol{\Omega}' = \sum_i \kappa_i' \mathbf{n}_i + \boldsymbol{\Omega} \times \boldsymbol{\Omega} = \mathbf{0}$, which means that *the Darboux vector is constant along each element*. For a given element Q, let us therefore introduce Ω the norm of the vector $\boldsymbol{\Omega}$ and $\boldsymbol{\omega} = \boldsymbol{\Omega}/\Omega$ the unit vector aligned with $\boldsymbol{\Omega}$ (the case $\boldsymbol{\Omega} = \mathbf{0}$ is considered separately, see below). Finally, we write $\mathbf{a}^{\parallel} = (\mathbf{a} \cdot \boldsymbol{\omega})\boldsymbol{\omega}$ and $\mathbf{a}^{\perp} = \mathbf{a} - \mathbf{a}^{\parallel}$ as the projection of an arbitrary vector $\mathbf{a}$ parallel to and perpendicular to the axis spanned by $\boldsymbol{\omega}$, respectively.

Since $\boldsymbol{\Omega}$ is constant, integration of equation (1.20b) over an element is straightforward. The material frame 'rotates' around $\boldsymbol{\omega}$ with a constant rate of rotation Ω per unit of curvilinear length. Therefore, the material frame at coordinate $s \in S_Q$ is obtained from the material frame $\mathbf{n}_{i,\mathrm{L}}^Q = \mathbf{n}_i(s_Q^{\mathrm{L}})$ given on the left-hand side of the interval S_Q, by a rotation with angle $\Omega\,(s - s_Q^{\mathrm{L}})$ and axis parallel to $\boldsymbol{\omega}$:

$$\mathbf{n}_i(s) = \mathbf{n}_{i,\mathrm{L}}^{Q\parallel} + \mathbf{n}_{i,\mathrm{L}}^{Q\perp} \cos(\Omega\,(s - s_{\mathrm{L}}^Q)) + \boldsymbol{\omega} \times \mathbf{n}_{i,\mathrm{L}}^{Q\perp} \sin(\Omega\,(s - s_{\mathrm{L}}^Q)). \tag{1.26a}$$

By equation (1.20a), the centerline $\mathbf{r}(s)$ is then found by spatial integration of $\mathbf{n}_0(s)$:

$$\mathbf{r}(s) = \mathbf{r}_{\mathrm{L}}^Q + \mathbf{n}_{0,\mathrm{L}}^{Q\parallel}(s - s_{\mathrm{L}}^Q) + \mathbf{n}_{0,\mathrm{L}}^{Q\perp} \frac{\sin(\Omega\,(s - s_{\mathrm{L}}^Q))}{\Omega} + \boldsymbol{\omega} \times \mathbf{n}_{0,\mathrm{L}}^{Q\perp} \frac{1 - \cos(\Omega\,(s - s_{\mathrm{L}}^Q))}{\Omega}, \tag{1.26b}$$

where $\mathbf{r}_{\mathrm{L}}^Q = \mathbf{r}(s_Q^{\mathrm{L}})$ is the prescribed position of the centerline on the left-hand side of the interval. Equations (1.26) provide the explicit reconstruction of an element. Its centerline is a helix with axis parallel to $\boldsymbol{\omega}$. An equivalent derivation based on Rodrigues' formula is given in [Pai02a]. Two degenerate cases are possible and must be considered separately: the curve is an arc of circle when $\tau = 0$ and $\kappa_1 \neq 0$ or $\kappa_2 \neq 0$; it is a straight line when $\kappa_1 = \kappa_2 = 0$, which can be twisted ($\tau \neq 0$) or untwisted ($\tau = 0$, implying $\boldsymbol{\Omega} = \mathbf{0}$).

Equations (1.26) can be used to propagate the centerline and the material frame from the left-hand side s_Q^{L} of the element to its right-hand side s_Q^{R}. The whole rod can then be reconstructed by applying this procedure over every element successively, starting from the scalp where $\mathbf{r}$ and $\mathbf{n}_i$ are prescribed by equation (1.20c). This yields explicit formulae for the functions $\mathbf{r}^{\mathrm{SH}}(s, \mathbf{q})$ and $\mathbf{n}_i^{\mathrm{SH}}(s, \mathbf{q})$, which have the form of equation (1.26) over each element. The integration constants are determined by continuity at the element boundaries.

1.4.4 Parameter values for natural hair

	Asian (smooth)	**Caucasian 1 (wavy)**	**Caucasian 2 (curly)**	**African (fuzzy)**
Radius (μm)	50	35	50	50
Ellipticity	1	1.1	1.1	1.2
Helix radius (cm)	0	1	0.6	0.1
Helix step (cm)	0	0.5	0.5	1
Young's mod. (GPa)	1	2	1.5	0.5
Poisson's ratio	0.48	0.48	0.48	0.48

Chapter 2

Hair Interactions

Florence Bertails, Basile Audoly, Marie-Paule Cani

2.1 Introduction

Human hair is a composite, deformable material made of more than 100 000 individual fibers called hair strands. As mentioned in the previous chapter, these thin tubular structures are elastic: after motion, they tend to come back to a rest shape, which is related to their individual natural curliness and to the set of external forces applied to them.

This chapter deals with the difficult problem of hair interactions, which plays a major role in the motion of a full head of hair, and even on the shape hair takes at rest: collisions and contacts between hair strands of different orientations cause hair to occupy a pretty high volume, especially in the case of irregular, curly or fuzzy hair. Due to this larger volume, tangled or fuzzy hair in motion is much more subject to air damping than smooth and disciplined hair.

The nature of interactions between hair strands is very complex. This is largely due to the surface of individual hair strands, which is not smooth but composed of tilted scales (see Figure 1.1, left). This irregular surface causes anisotropic friction inside hair, with an amplitude that strongly depends on the orientation of the scales and of the direction of motion [Zvi86]. Moreover, hair is very triboelectric, meaning it can easily release static charges by mere friction. This phenomenon,

which has been measured in the case of combed hair, most probably impacts the hair-hair friction rates.

Because of the extremely large number of strands that compose a full head of hair, processing hair interactions is known as one of the main challenges in hair animation. Until the late nineties, most hair animation methods tackled hair collisions with the body, but were not processing self-interactions at all. This often resulted into an obvious lack of hair volume. The first methods that detected interactions between hair wisps spent more than 80% of the simulation time in this process. More recently, several interesting solutions that make hair interactions much more practical were developed: some of them mimic the effect of hair interactions globally, using a structure that stores the volumetric density of hair. Others achieve more accurate results by developing efficient algorithms for detecting collisions between hair-wisps and by setting up realistic models for response and friction forces.

This chapter presents those of these recent advances in which the authors participated: Section 2.2 briefly reviews the two main approaches for animating hair, namely modeling hair as a continuum or as a set of individual hair wisps. The associated methods for processing hair interactions with the body are presented and the issues raised by hair self-interactions are introduced. Section 2.3 presents a practical real-time solution, applicable in any hair animation system, which gives hair a volumetric behavior without requiring to detect individual interactions between the animated guide-strands. We then focus on more accurate methods, applicable for generating high quality animation of long hair: Section 2.4 reviews some recent methods for efficiently, yet robustly detecting the interactions between guide-strands. Section 2.5 discusses the anisotropic models that were set up to model response to these interactions. In particular, we describe a validated model for friction forces. In conclusion, we emphasize the steps forwards made in the last few years, but also the issues that were not tackled yet, showing that improving the efficiency and visual realism of hair animation is going to remain a hot research topic for a while.

2.2 Hair animation and interaction processing

2.2.1 Continuous versus wisp-based hair models

Hair animation was made practical in the early nineties [RCT91b] by the idea of animating only a sub-set of the hair strands (typically one or two hundreds), which we will call here the *guide-strands*. This is made possible by the local spatial coherence of hair motion. Once the guide-strands have been animated (using for instance spring and masses, projective dynamics or chains of articulated rigid bodies), their position is used to generate the remaining hair strands at the rendering stage.

More precisely, two main families of approaches were developed for modeling hair: The first ones, more appropriate for smooth, fluid hair, consider hair as a continuum [AUK92b, DMTKT93, HMT01a, CJY02a, BCN03] and thus use interpolation between the animated guide-strands for generating a full head of hair. The second ones, which achieve their best results for wavy of curly hair, model hair as a set of disjoint wisps [CSDI99, KN00, PCP01a, KH01, BKCN03a, WL03, CCK05a]. The animated guide-strands are assimilated to wisp skeletons and extrapolation is used for generating extra hair-strands within each wisp. Recently, Bertails [BAC$^+$06] bridged the gap between the two kinds of approaches by allowing the guide-strands to be used both for interpolation or approximation depending on the type of hair and on the current distance between neighboring guide-strands. This model captures hair that looks like a continuum near the head while well identified wisps can be observed at the tip.

In the remainder of this chapter, we will discuss hair interactions independently of the hair model used among the approaches above: hair will be considered as a set of individual hair guides, each of them more or less explicitly modeling a volume of hair around it. Interactions will be detected and treated based on the position and motion of these guide-strands.

2.2.2 Processing hair interactions with the body

The first step towards processing hair interactions is to adequately model hair collisions and contacts with obstacles, starting with the body of the animated character. Since hair is animated using guide-strands, the latter and the wisp volumes

around them (if any) should be prevented from penetrating inside the body. The latter is often approximated using sets of ellipsoids or stored in a spatial partitioning grid to accelerate this detection. Since hair is a very soft material, modeling a one way response is sufficient: the body can be considered as infinitely rigid and heavy compared with hair, so the collision has no effect on the subsequent body shape and motion. Moreover, hair is a very soft and light material: it does not bounce after collision, but rather experiment a strong static friction with the parts of the body it is in contact with. Collision response can thus be treated using methods set up for other very light material, such as clothing: when a penetration is detected, the guide-strand or the associated wisp volume is re-positioned as to be in resting contact with the body. The guide-strand is either given the velocity of this body part, or a static friction force is set up between them.

The remainder of the chapter focuses on the part of interaction processing most specific to hair and much more difficult to handle than collisions with obstacles: we are now addressing the challenging problem of self-interactions.

2.2.3 The issues raised by hair self-interactions

The interactions that occur between hair-strands are very difficult to simulate, For the following reasons:

Firstly, in real hair, the friction between neighboring strands of similar orientation plays an important part: it dissipates some kinetic energy and damps the overall motion. This phenomenon cannot be simulated properly in virtual hair, where only a few guide-hair distributed on the scalp are animated. The only way to capture this part of self-interaction is to add some internal damping - which should depend on the type of hair and is quite difficult to tune - on the individual motion of a guide strand.

Secondly, strands are very thin, so standard collision detection methods based on penetration cannot be used: strands or even small wisps of hair of different orientations might cross each other between two simulations steps and go to rest in the wrong positions, this interaction remaining un-noticed.

Lastly, once a collision between hair guides or hair wisps of different orientation have been detected, the response model should account for the complex state of surface of a hair strand: the tilted scales that cover a strand result in strongly anisotropic static friction. Moreover, these friction forces are dominant: due to

the lightness on a hair strand, the colliding strands will most probably remain in contact. One of the challenges of hair self-interactions it thus to define a response model that prevents strands from crossing each other while avoiding to generate any bouncing. The latter, often noticeable in hair animation systems, gives an overall unstable behavior to the full hair, due to the extremely large number of local collisions that occur at each time step, even when hair is at rest.

Historically, the continuous and wisp-based approaches have tackled hair self-interactions in dramatically different ways:

- **Volumetric interactions:** Continuum approaches such as Hadap's and Bando's methods relied on fluid-like internal viscosity to model hair friction and to prevent self-intersections is a rather global way [HMT01a, BCN03]: no collision is detected between individual hair strands, but the latter interact (as fluid particles would do), depending on the local hair density and on the relative hair motion around them.

- **Guide-strands interactions:** In contrast, processing hair self-collision in discontinuous, wisp-based approaches has been done through the actual detection of penetration between moving hair wisps [PCP01a]. This allows a more accurate modeling of the discontinuities that can be observed during fast motion of long, human hair: in these approaches, wisps of hair defined around a guide-strand are prevented from crossing each other and two wisps of different orientations can be in resting contact.

We believe that the general approach chosen for handling hair interactions can be chosen quite independently from the hair model, would it be a continuum model, an disjoint set of hair wisps, or something inbetween.

The remainder of this chapter presents the specific solution the authors have developed for tackling the problem of hair interactions. This chapter is not aimed as providing a state of the art in the area: the interested reader can find a recent survey on hair animation and rendering techniques in [WBK$^+$07]. The volumetric method for hair interactions presented in Section 2.3 belongs to the volumetric interactions approach: it provides a real-time alternative to fluid-like interactions when a coarser approximation is sufficient. Methods for improving the efficiency of collision detection and the realism of collision response in the interacting guide-strands approach are detailed in Sections 2.4 and 2.5.

2.3 A volumetric approach for real-time self-interactions

The work presented in this section was first introduced in [BMC05], as a side application of a method for handling hair self-shadowing in real-time. We detail here the application of this approach to hair self-interactions.

2.3.1 A volumetric structure for hair

An acceptable approximation of hair self-interaction consists of considering that internal collisions mainly result into the preservation of hair volume [LK01a]. Starting from this assumption, hair density information is very useful: If the local density of hair is over a fixed threshold (corresponding to the maximum quantity of hair that can be contained within a cell), the hair strands should undergo external forces that spread them out.

Bertails *et al.* [BMC05] use a light-oriented voxel grid to store hair density values. This enables them to efficiently compute *both lighting and mechanical interactions* inside the hair volume in real-time. Though very simple, this method yields convincing interactive results for animated hair, is very simple to implement, efficient and can easily be parallelized to increase performance.

More precisely, the volumetric structure used is based on a 3D light-oriented density map, which combines an optimized volumetric representation of hair with a light-oriented partition of space. This voxel structure stores the local hair density in space, computed from the number of guide-strand segments within a given cell. It is used to approximate the light attenuation through each cell of the grid: since the cells are sorted along the light direction, computing the accumulated translucency for each cell through the hair volume becomes straightforward.

2.3.2 Application to hair interaction

At each animation step, all guide-strand are moved to their new position and the density map is updated. Then, hair self-collisions are taken into account for the next simulation step by adding density-based interaction forces where needed: repulsive forces directed from the center to the border of a grid cell are generated.

They are applied to each hair-guide element located in a cell whose density if over a threshold. This threshold value depends on the desired level of hair fuzziness.

Although this interaction method is extremely simple, it yields convincing results. In practice, it was tested with an accordingly simple, yet robust algorithm for animating the guide-strands: hair is composed of approximately a hundred wisps, each of which being simulated using three guide-strands modeled as chains of rigid links. The latter are animated using a fast and robust but non-accurate method [Ove91]. The rendering technique is a hybrid between continuum and wisp-based methods: interpolation between the three guide-strands is used to generate a continuum of hair inside each deformable wisps. The overall method results into interactive hair animations that include self-interactions as well as self-shadowing, and generate visually convincing hair volume (see Figure 4.6). Furthermore, with this technique, handling hair self-collisions only requires 2.5% of the whole processing time.

Figure 2.1: Interactive hair self-shadowing processed by accumulating transmittance values through a light-oriented voxel grid [BMC05]. (left) Animated smooth hair; (right) Animated curly hair.

2.4 Detecting guide-strand interactions

Volumetric methods as the simple solution presented above are not sufficient for generating high quality animation of non-smooth hair: two hair wisps of different

orientations may cross each other during motion despite of the volumetric forces they undergo. Most hair animation methods have thus relied on the distance between pairs of guide-strands or on the penetration between wisps of hair defined around them for accurately detecting hair self-interactions. In this chapter, we call these more accurate approaches *guide-strand interactions*.

A naive implementation of guide-strand interactions would lead to $O(n^2)$ tests, where n is the total number of guide-strand segments (or wisp segments) in the hair model. Following Plante [PCP01a], most methods use a pre-detection based on a regular 3D grid data structure, built around the character and its hair, to quickly get rid of most non-intersecting cases. Each grid cell contains a list of hair-guide elements (or wisp segments) whose bounding box intersects the cell. At each animation step, the grid is used for quickly determining a shorter list of segments susceptible to intersect. A mailbox parameter indicates the last time step when a given pair of such segments has been tested, ensuring that each pair is tested only once. The 3D grid data structure can also be used for optimizing collision detection between hair and the character model: to achieve this, each cell also references the polygons of the character model that intersect it.

2.4.1 Deformable versus cylindrical hair wisps

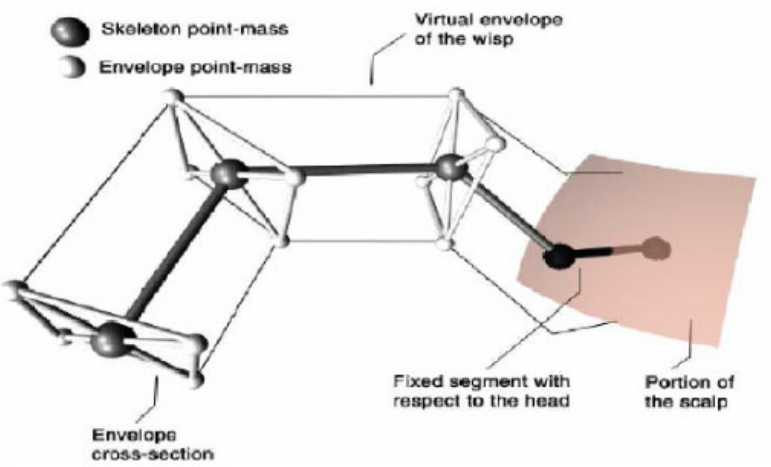

Figure 2.2: Elements defining a deformable volumetric wisp [PCP01a].

To account for the complex interactions observed in real hair during fast motion, Plante *et al.* represented hair using a fixed set of deformable, volumetric wisps [PCP01a, PCP02]. Each wisp is structured into three hierarchical layers: a skeleton curve (called here guide-strand) that defines its large-scale motion and deformation, a deformable volumetric envelope that coats the skeleton and accounts for the deformation due to hair interaction within a wisp, and a given number of hair strands distributed inside the wisp envelope and which are generated only at the rendering stage (see Figure 2.2). More precisely, the deformable sections that shape a wisp of hair around its guide-strand are animated using 4 1D damped springs, attempting to capture the way a wisp of hair deforms when its moves and most often comes back to its initial size at rest. The wisp volume was defined as a quadratic surface envelop controlled by these cross-sections.

Using such a complex deformable wisp model for the detection of guide-strand interactions proved very time consuming: more than 70% of the simulation time was used in collision detection between hair wisps, despite of the space grid used to accelerate the process. In total, without taking hair rendering into account, about 3 hours of computations were required, in 2001, to compute 3 seconds of animation.

Bertails *et al.* [BKCN03a] introduced an adaptive animation control structure, called the *Adaptive Wisp Tree* (AWT), which enables the dynamic splitting and merging of hair wisps. The AWT depends on a full hierachical structure for the hair, which can either be precomputed - for instance using a hierarchical hairstyle [KN02] - or computed on the fly. The AWT represents at each time step the wisps segments (or guide-strand segments) of the hierarchy that are actually simulated (called *active* segments). Considering that hair should always be more refined near the tips than near the roots, the AWT dynamically splits or merges hair wisps while always preserving a tree-like structure, in which the root coincides with the hair roots and the leaves stand for the hair tips.

In addition to limiting the number of active hair-wisp segments, one of the key benefits of the AWT for collision detection is that the splitting behavior of the wisps models their deformation: there is no need for the complex deformable wisp geometry used in [PCP01a]. For collision processing, active wisp segments of the AWT are thus represented by cylinders, which greatly simplifies collision detection tests: detecting interactions simplifies into detecting the local minima of the distance between guide-strand and comparing its value to the sum of the wisp radii. With this method, ten seconds of animations could be computed, in 2003,

in less than five minutes.

2.4.2 Handling curly hair and exploiting temporal coherence

The Super-Helix model, which was recently introduced at SIGGRAPH [BAC+06], and presented in chapter 1, is the first model that accurately simulates the dynamics of curly hair: unlike previous approaches, curly hair wisps are not modeled using a straight mass-spring skeleton around which wavy strands are drawn at the rendering stage, but are instead accurately modeled using wavy to fuzzy guide-strands, which have a piece-wise helical shape. Detecting interactions between such complex helical guide-strands is indeed more costly.

To handle collisions between hair clumps guided by Super-Helices in a both accurate and efficient way, our strategy is based on the two following ideas: 1) the use of *adaptive* cylindrical bounding envelopes around each hair wisp, whose number and size can automatically adapt during motion, depending on the geometry of the wisp, and 2) the *tracking of the closest points* between the skeletons (*i.e.*, the principal axes) of the bounding cylinders.

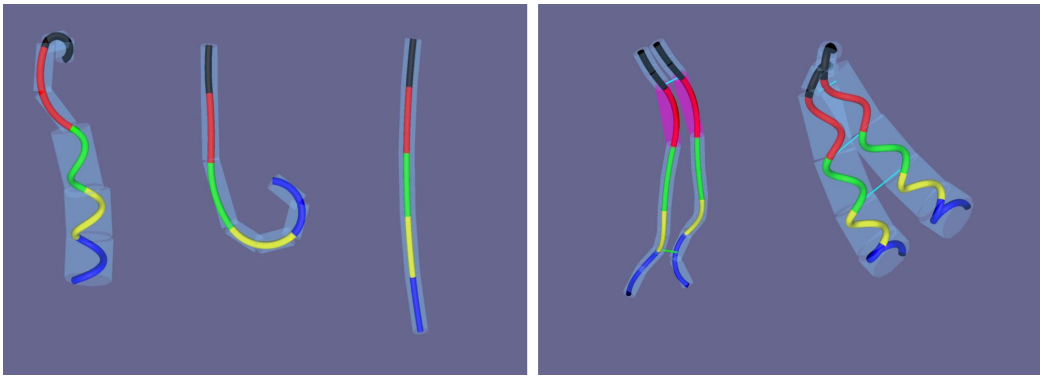

Figure 2.3: *Left: The three different adaptive representations for the bounding volume of a wisp segment. Right: Tracking the pairs of closest points between the skeletons of guide volumes (for smooth and curly hair) [Ber06].*

1. **Adaptive bounding envelopes**: the bounding volume of a helical element Q_i of the guide hair strand is composed of a single, large cylinder if the helix's spires are tight enough. In other cases (*i.e.* for straighter strands),

we use one or two cylinders, oriented along the mean local tangent of the element, to approximate the volume of the wisp (see Figure 2.3).

2. **Tracking pairs of the closest points**: we adapted the algorithm of Raghupathi *et al.*, originally designed for detecting self-collisions in long and thin deformable objects [RCFC03], to the collision detection between guide hair volumes. Since guide hair volumes are composed of a set of cylinders, the method amounts to computing minimal distances between pairs of segments (the principal axes of the cylinders), as in [RCFC03]. For each pair of guide-strands, we first initialize a closest point pair near the root. At each time step, each closest point pair is updated by letting the closest points slide along the associated wisp, from the positions they had at the last time step. They stop in a location that locally minimizes the distance between the two wisp volumes. When this distance is under a threshold, new pairs of points are created at both sides of the initial pair, to track the possible multiple local minima. When two closest point pairs slide to the same location, they are merged together. At each time step, because of temporal coherence, only very few of these pairs need to be moved, so advancing them is very fast. Each time the distance between two guide volumes is locally smaller than the sum of their radii, collision is detected.

This algorithm ensures that at least one pair of closest points is maintained between two guide volumes, while keeping the number of tracked pairs between guide volumes low (merging occurs when two different pairs slide towards the same place). The algorithm has thus a n^2 complexity where n is the number of guide hair strands composing the hairstyle instead of the total number of segments composing hair, as it would be when using a naive algorithm.

The same adaptive wisp volumes and temporal coherence technique are used for detecting collisions between the hair and the body of the character. Distance tests are computed between segments and spheres, as the body is approximated by a unions of spheres. Using this technique, we obtained a total frame rate of only 3 seconds per frame for a dynamic hair style composed of a hundred of guide hair strands, including self-interactions and interactions with the body.

2.5 Response to guide-strand interactions

As already mentioned hair is a very soft and light material. Seen as a whole, it deforms rather than bouncing when it collides with a relatively rigid obstacle such as the character's body. Indeed, hair self-collisions should be very soft as well, and result into frictional rather than bouncing behaviors. Therefore, response to guide-strands interactions have been modeled using soft penalty forces together with friction forces.

2.5.1 Anisotropic response in wisp-based methods

As noted by Plante *et al.* [PCP01a, PCP02], accounting for collisions between hair wisps is fairly different from modelling collisions between standard deformable bodies. Wisps are highly anisotropic, since they are just a virtual representation for a group of hair strands. While two perpendicular colliding wisps should be compressed in order to avoid intersection, interpenetration can be allowed between neighbouring wisps moving roughly in the same plane. In consequence, the authors proposed an anisotropic model for the interactions between hair wisps: Wisps of similar orientations are mostly submitted to viscous friction and penetrate each other, whereas wisps of different orientations actually collide in a very dissipative way.

Figure 2.4: The layered wisp model [PCP01a] (right) captures both continuities and discontinuities observed in real long hair motion (left).

As illustrated in Figure 2.4, this approach yields convincing results, even for fast motions: the model adequately captures the discontinuities that can be observed in

long, thick hair, preserves hair volume and prevents crossing between hair wisps. Nevertheless, the high number of contacts that needed to be computed between the different wisps at rest caused some noticeable artifacts such as unstabilities when hair comes to rest.

The previous anisotropic collision response model was re-used and improved by the Adaptive Wisp Tree (AWT) method [BKCN03a]: an AWT implicitly models some of the mutual hair interactions, since neighboring wisps with similar motions merge, thus efficiently yet robustly mimicking the static friction in real hair. This merging behavior also avoids subsequent collision processing between these wisps, thus increasing efficiency as well as gaining stability from the reduced number of primitives. Typically, an AWT simulation starts with a reduced number of hair wisps. While the character moves, these wisps refine where and when needed (see Figure 2.5), to merge again as soon as they can. When the character is back at rest, the simulation eventually ends up a single large hair wisps. This totally avoids the local unstabilities noted in previous approaches.

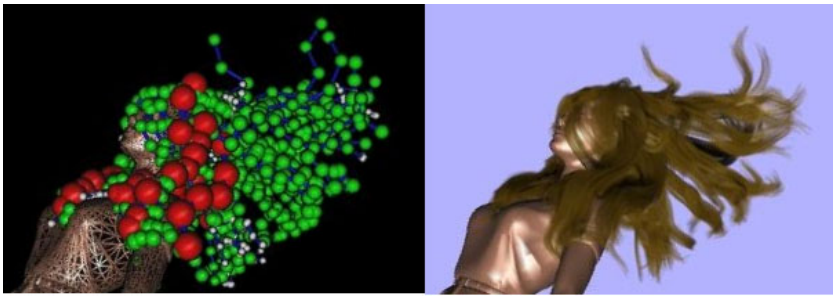

Figure 2.5: Left: The structure of an AWT at a given animation step. Most of the parent wisps (in red) have split into medium-size wisps (in green), which eventually have split into small ones (in white). Right: Rendering of the same frame [BKCN03a].

2.5.2 Setting up realistic penalty and friction forces

The recent work on Super-Helices tackled the problem of setting up more accurate response forces between interacting guide-strands [BAC+06]. Interactions between guide-hairs, and between hair and external objects (such as the body) are performed through penalty forces which include a normal elastic response together with a tangential friction force.

As in [Dur04], the normal penalty force is stabilized thanks to a quadratic regularization for small penetrations. From a regularization depth δ_{reg} (arbitrarily chosen), the normal reaction force $\boldsymbol{R_N}$ exerted between the two closest points of interacting guide-strands is computed as follows:

$$\begin{cases} \text{if } (gap \leq 0) & \boldsymbol{R_N} = \mathbf{0} \\ \text{if } (0 \leq gap \leq \delta_{reg}) & \boldsymbol{R_N} = \frac{k_c\, gap^2}{2\,\delta_{reg}} \boldsymbol{n_c} \\ \text{else} & \boldsymbol{R_N} = k_c\,(gap - \frac{\delta_{reg}}{2}) \boldsymbol{n_c} \end{cases}$$

where $\boldsymbol{n_c}$ is the unitary vector giving the direction of collision (calculated as the cross product of the vectors defining the two closest segments), and k_c an arbitrary constant value.

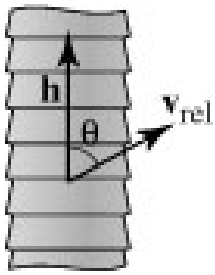

Figure 2.6: *Angle θ between the fiber orientation and its relative velocity w.r.t the external object in contact with the fiber.*

To simulated friction between wisps in contact or friction with an obstacle, the method extends viscous friction law in [CK05], defined as :

$$\mathbf{R_T} = -\nu\,(\mathbf{v_{rel}} - (\mathbf{v_{rel}}.\mathbf{n_c})\,\mathbf{n_c})$$

To account for the oriented scales covering individual hair fibers, the friction coefficient ν is multiplied by a sine function to account for the orientation of hair fibers with respect to their sliding motion over the external object: $\nu = \nu_0\,(1+\sin(\theta/2))$, where angle θ is defined in Figure 2.6.

The parameters of interaction forces, as well as the other parameters of the Super-Helices model, can be set up using the actual study of real wisps of hair: The friction parameter ν_0 between hair and a given material is directly adjusted from real observations of sliding contacts between the hair clump and the material.

As Figures 2.7 and 1.10 show, the Super-Helices model results in realistic simulations which can be compared side by side with videos of real hair in motion.

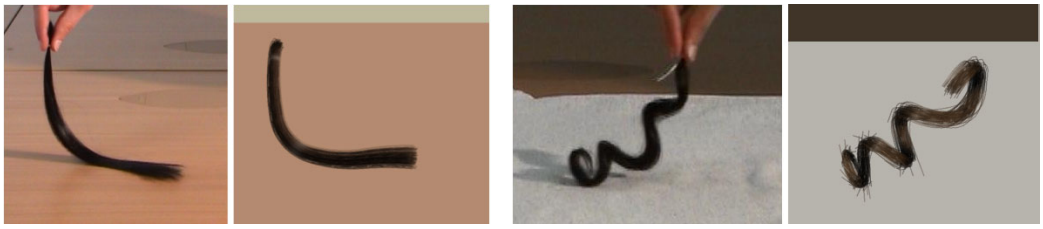

Figure 2.7: *Validation of the friction model in [BAC+06] on a sliding motion of a smooth (left) and curly (right) hair clump over different kinds of material (left: wooden surface, right: cotton).*

2.6 Conclusion

As we have shown, processing hair interactions requires a dedicated set of methods, due to the very specific nature of the hair material. Impressive advances were made in the last six years, from the first models able to handle hair self-collisions to efficient, robust and even partly validated methods. This chapter has detailed several specific solutions that range from the use of a volumetric approach when a very quick solution is required to realistic models that still keep the computational load to an acceptable rate.

In spite of all these advances, there still remains very challenging issues in the modeling of hair self-interactions: these interactions are indeed the origin of the complex collective behavior of hair. Especially they cause hair to group into clusters during motion; this phenomenon has never been accounted before (except in very simplified models, such as the AWT), as previous models usually assume that hair granularity is fixed by the number of simulated guide-strands. Moreover, hair interactions vary a lot according to external conditions such as moisture (wet hair being the extreme case), combing, or the use of cosmetic products. Lastly, hair tribo-electricity has never been modelled in an accurate way.

Future research should include attempts to make volumetric methods such as the one presented in section 2.3 more accurate at low cost, by taking local hair directional distribution into account while setting up the response force. The approaches that seek for realism should probably extract the internal damping inside a hair wisp from the preliminary study of hair chunks actually modeled using a full set of interacting hair strands. This study should also bring more accurate criteria for splitting a wisp into sub-wisps or merging them, and could help char-

acterizing the number of hair guides required according to the natural curliness and smoothness of a given hair type.

Chapter 3

Multi-Resolution Hair Modeling

In this chapter, we present the basic framework for level-of-detail hair modeling. These methods determine on the fly which hairs are of most significance to the simulation and provide novel techniques to allocate the majority of the computational resources towards modeling these hairs. This process then accelerates the simulation of hairs deemed less important, thereby accelerating the overall hair simulation while maintaining the desired visual quality of the total simulated hair.

Traditional hair modeling techniques have viewed hair in different manners. Hair is typically seen as individual strands, or one-dimensional curves in three-dimensional space. Sometimes, hair is modeled as groups of strands, or *wisps*, where multiple rendered strands were animated and styled as larger groups. These disjoint hair groups are also modeled as strips of hair through two-dimensional surfaces. Hair at times is also perceived as one large volume; animation and styling is controlled through a continuous medium while either one-dimensional strands or surfaces can be used to render the volume of hair.

These separate hair modeling representations have typically mandated a choice between simulation quality and simulation speed. The impetus of this research is to dynamically create a balance between quality and speed for hair modeling. To attain this goal, it is necessary to allow the hair model to adapt to the changing simulation, finding the balance between simulation speed and simulation quality. In this chapter, we will describe the three representations we use to model hair. These representations, which we also refer to as the *discrete levels-of-detail* for modeling hair include *individual strands*, *clusters*, and *strips*, see Figure 3.1.

The *individual strands* provide the finest level-of-detail, are modeled with one-dimensional subdivision curves, and can be grouped to follow traditional wisp animation schemes. The *clusters* are a new representation formed from generalized swept volumes created with subdivision surfaces to model a volume of hair. The *strips* are the lowest level-of-detail and are created from flat two-dimensional subdivision surfaces. These representations were first introduced by Ward *et al.* [WLL+03].

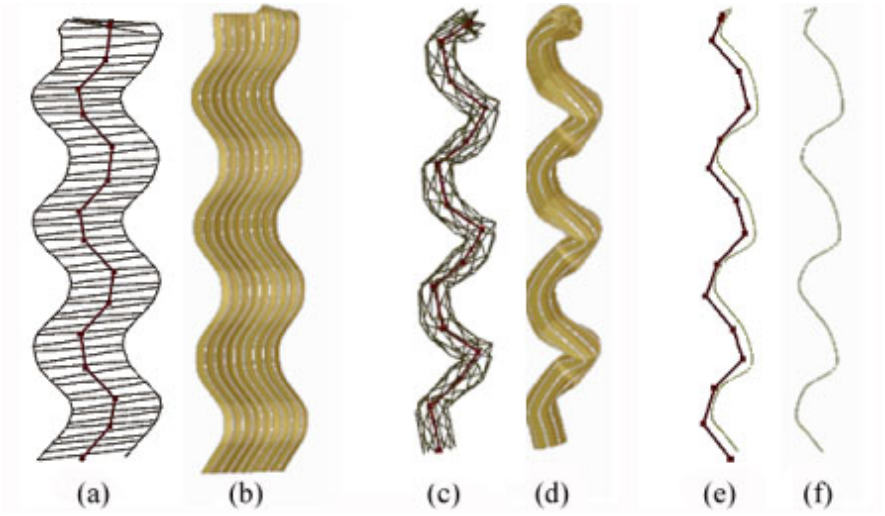

Figure 3.1: Level-of-Detail Representations for Hair Modeling. *(a) Subdivision representation of strip with skeleton; (b) Rendered strip; (c) Subdivision representation of cluster with skeleton; (d) Rendered cluster; (e) Subdivision representation of a strand with skeleton; (f) Rendered individual strand.*

This chapter will also introduce the base skeleton used to control the motion and shape of each level-of-detail. The base skeleton is the underlying control structure for each LOD representation and dictates the placement of control vertices used for subdivision. The base skeleton plays an important role in level-of-detail transitioning; it is intentionally selected to maintain a global, consistent, macroscopic physical behavior as LOD switches take place. Using the same base control structure for each LOD helps to drastically simplify many transition difficulties typically present during LOD switching. It automatically reduces a fairly high degree-of-freedom dynamical system down to a lower degree-of-freedom dynamical system without any extra expensive computations other than performing the LOD switching tests.

Hair simulation is controlled through the use of the base skeleton. We introduce a collision detection method that efficiently and correctly handles hair-object and hair-hair interactions.

We will also explain how the different framework entities work together to model hair and show how the LOD framework can model hair more efficiently than previous methods, while maintaining a high visual quality. We have developed methods for choosing the appropriate LOD for modeling a section of hair based on a number of criteria, including visibility, viewing distance, and hair motion. In this chapter, we will discuss how these criteria are used together to choose the final representation for hair, transition between the different representations, and show results and discuss the performance of these methods.

3.1 Geometric Representations

Using a base skeleton, the three discrete hair representations can then be created. Each LOD representation has varying simulation complexity and visual fidelity for modeling hair and they have been chosen to be used together due to these variations.

3.1.1 The Base Skeleton

The base skeleton controls both the shape and the motion of each hair representation at any given point during simulation. Based on the idea for modeling each individual hair strand [AUK92b, KAT93], a structure has been employed for the base skeleton that forms the ”core” of the proposed set of LOD representations. This base skeleton is comprised of n control points, or nodes. This value is decided based on criteria involving the length of the hair, the waviness or curliness specified for the hair, and the desired smoothness for motion. The higher the number of control points, the higher the complexity of the system and the finer the detail is. The skeleton is modeled as an open chain of rigid line segments that connect these nodes. The shape of the skeleton is controlled by polar coordinate angles between each node. The Eulerian distance between each node is fixed, thus preventing the length of the hair from changing during simulation.

3.1.2 Strips

The strip model in Figure 3.1(a) and (b) uses a single base skeleton model as its foundation for motion. The structure for this model is inspired by the strips representation presented by [KH00, KH01]. The skeleton is the center of the strip and for each node in the skeleton there are two control points that are used to define the strip. These two strip control points and the skeleton node point are collinear. A skeleton with n nodes will result in a subdivision surface created from a control polygon consisting of $2n$ control points.

A strip is typically used to represent the inner most layers of hair or parts of hair that are *not fully visible* to the viewer and, therefore, are often not rendered. It is the coarsest (lowest) level-of-detail used for modeling hair. It is mainly used to maintain the global physical behavior and the volume of the hair during the simulation.

While the strip representation gives better visual results for straight hair, it can also be used to model wavy and curly hair, but not in as fine a detail as the clusters or strands. Strips are only used when the viewer cannot observe fine detail, such as when the hair is at distances far from the viewer, or when the hair is not in sight. Thus, while the strip cannot depict all hairstyles as accurately as the other two LODs, it is typically not visible to the viewer. Criteria for choosing an LOD is discussed in further detail in Section 3.3.

3.1.3 Clusters

The clusters are represented as generalized cylinders created with texture-mapped subdivision surfaces, as shown in Figure 3.1(c) and (d). Each cluster is formed from one skeleton that is located at the center of the cluster. A radius is specified at the top and the bottom of each cluster. The radius is then linearly interpolated at each skeleton node point; this allows the thickness to vary down the length of the cluster. At each skeleton node, a circular cross-section, made up of m control points, is created based on the radius value at that node. Thus, a skeleton made up of n points will create a cluster of mn control points. Typically having m=4 is enough detail to define the cross-section.

A cluster is used to model the intermediate layers of hair and often makes up the majority of the body of semi-visible hair. Whenever appropriate, it is far

less costly to represent a group of hair using the cluster model, instead of a large number of individual strands. The cluster is able to give more detail than the strip representation because it more accurately represents a given *volume* of hair since it is rendered as a textured cylindrical surface. However, the cluster requires more control points than the strip making the complexity to both simulate and render it more costly. A single cluster though can approximate a large number of strands, considerably decreasing the number of base skeletons required for simulation and the number of control points for rendering in comparison to strands alone.

3.1.4 Strands

Each individual strand is modeled as a subdivision curve using 1D subdivision with n control points, as shown in Figure 3.1(e) and (f). A single control vertex is created for each node in the skeleton. Strands capture the most detail in comparison to the other representations; nevertheless they also require the most computation. Multiple strands are grouped to follow the same skeleton to create strand groups or wisps. This process captures many realistic behaviors of hair since real hair tends to group together due to static electricity, oils in the hair, or other substances in the hair such as water or styling products. The observation that hair strands near each other behave similarly allow for a more efficient modeling of individual strands. Still, these groups of strands are more expensive to simulate than the clusters or strip representations; moreover each strand is still rendered making the strands more costly for rendering in comparison to clusters and strips. A strand group containing j strands will then comprise jn control vertices before subdivision.

3.2 Hair Hierarchy

The previous sections introduced the basic components for level-of-detail hair modeling. In this section, we introduce the *hair hierarchy*, a control structure that provides further refinement to the LOD hair framework. The hair hierarchy increases control over the resolution as it contains various numbers and sizes of each discrete representation. The hair hierarchy was first presented by Ward and Lin [WL03].

Using the hair hierarchy the coarsest representation for a given volume of hair is still a single strip. To gain more resolution, however, the hair hierarchy allows the volume to transition into multiple smaller strips before reaching the cluster level. Likewise, in cluster form, the volume of hair can now be represented with various numbers of clusters that differ in size as well as visual fidelity and performance speed. As the number of clusters that are used to model a volume of hair increases so does the visual fidelity of the simulation. Finally, rather than using groups of strands of static sizes, the hair hierarchy allows these strand groupings to merge and split on-the-fly, simplifying or adding detail to the simulation in the process.

The hair hierarchy is created through the continual subdivision of strips, clusters, and strand groups and upon completion, contains varying resolutions of each discrete representation. As a result, near continuous level-of-detail control over the simulation is provided. A hair hierarchy is traversed on-the-fly during the simulation to not only select the appropriate discrete representations for a section of hair, but also the appropriate resolutions of the representations.

In addition to providing further level-of-detail control, the hair hierarchy actually captures a behavior of hair that numerous hair modeling techniques ignore; this effect is the dynamic clustering of hair strands often exhibited in long hair. While strands of hair in close proximity with each other do tend to follow similar motions (an underlying assumption of most hair modeling techniques), strands can often collect into large disjoint groups of strands that remain separate from the majority of the hair volume (a property continuum-based approaches often lack). These large disjoint groups of strands can actually break into smaller groups, or strands can leave one group and join another group under large motions, a behavior referred to as *dynamic clustering*, which static wisp-based approaches fail to capture. The hair hierarchy can simulate dynamic clustering effects as it simulates groups of hairs split and merge as simulation factors change.

In this section, we explain the construction and storage of the hair hierarchy, which is performed as a pre-process to the simulation.

3.2.1 Strip and Cluster Subdivision

Before a hierarchy of strips or clusters can be built, the initial top-level strip must be created. A top-level strip is created by choosing a location on the scalp for the origin of the skeleton (the first node point of the skeleton). Next, a user-defined

width is specified controlling the thickness of the initial strip.

Because the strip is a two-dimensional surface, its subdivision is restricted such that it may only be split into two equal parts. Strip subdivision is simply the degenerate case to cluster or strand group subdivision, using a degenerate quad-tree, or a binary tree, instead of the quad-tree data structure that is used for cluster and strand group hierarchies. The subdivision ends once the width of the current strip is below a user-defined threshold; these strips then become the leaves of the strip hierarchy.

To create the cluster hierarchies, leaf strips are divided into two equal-sized clusters, which become the root clusters of the cluster hierarchies. The cluster subdivision starts with the circular cross-section that defines the cluster. This circular cross-section is then split into four equal parts. The four sub-clusters have the same radius value but represent four different quadrants of the original cluster. The subdivision of a cluster always results in four children, so its information is held in a quad-tree. Clusters stop subdividing once their radius is below a user-defined threshold value. At this point, further detail is created in the strand group hierarchies.

3.2.2 Strand Group Subdivision

A strand group cross-section is illustrated in Figure 3.2a. The individual hair strands are randomly placed within the group and follow the dynamics of the skeleton. The circular shape of the strand groups is used for its simplicity in collision detection.

A quad-tree data structure contains the hierarchy information. It follows therefore, that each strand group is split into four equal sections, as shown in Figure 3.2b. The subdivision of a strand group into four sections creates a tight fitting circular cross-section for each subgroup, as in Figure 3.2c and Figure 3.2d.

Once the strand group is divided, the number of strands in each quadrant is calculated. If a quadrant has no strands within its boundaries then the child associated with that quadrant is set to null (see Figure 3.2e). A strand group will have between zero and four children. A strand group that contains only one strand will have zero children and becomes a leaf in the tree. It may not be necessary for the strand hierarchies to reach the individual strands in a simulation if the user

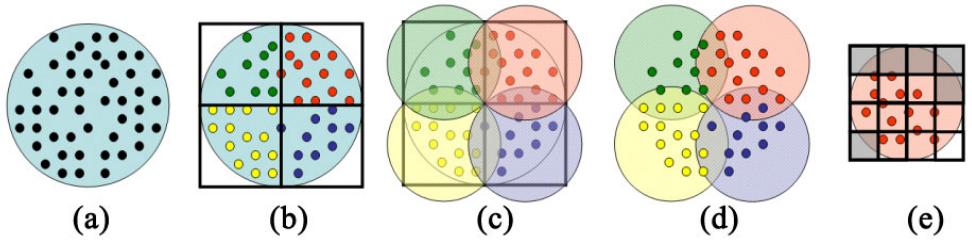

Figure 3.2: Strand group subdivision. *The subdivision process of a strand group into multiple strand groups. (a) The cross-section of a single strand group. (b) Strand group is divided into 4 equal quadrants and the strands are separated by the quadrant in which they lie (designated by different colors). (c) Circular cross-section is fit around each quadrant, or child, of original strand grouping. (d) Four new strand groups are created which are children of the original strand group. (e) Continual subdivision process is repeated on each child. Tinted squares show empty quadrants that contain no strands, these quadrants are set to null.*

does not desire that much detail. In that case, as an alternative the user can decide a minimum number of strands in a group. When a strand group contains the minimum number, or less, the subdivision stops.

3.3 Runtime Selection of Hair

Now that the level-of-detail framework for hair has been discussed, this section will explain how they work together to accelerate the simulation and rendering of hair. The primary goal of this framework is to measure the importance of a section of hair to the application and use that importance to determine the appropriate LOD to model the section of hair. At any given time during the simulation, a head of hair can be comprised of an assortment of LOD representations, meaning strands, clusters, and strips are used together to balance the visual quality and simulation performance for a head of hair. Using this method, the majority of computational resources are used to model the hair that is most significant to the application.

The importance of a section of hair is measured based on how much detail there is for the viewer to observe. The less observable detail there is for a section of hair, then the less important it is deemed for the viewer and it is then simulated

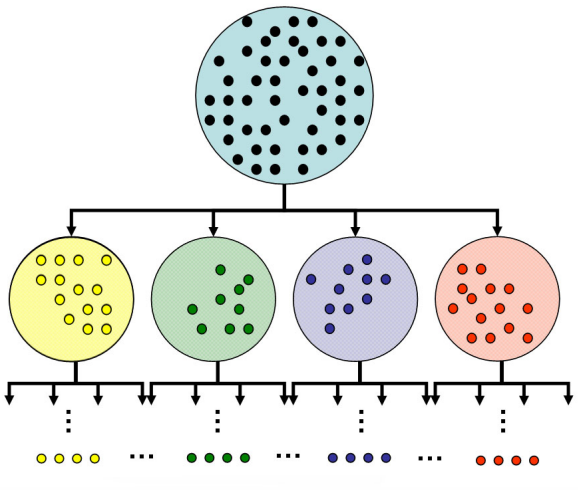

Figure 3.3: Strand group hierarchy. *Subdivision process creates a quad-tree containing strand group information. Strand group hierarchy can extend to individual strands.*

and rendered with a coarser LOD. A *section* of hair, in this context, is defined to be a given volume of hair that can be modeled using any level of the created hair hierarchy.

We have developed three criteria that measure the importance of a section of hair to the application. These criteria include the following:

- **Visibility** - Measures if the viewer can see the section of hair;
- **Viewing distance** - Measures how far the section of hair is from the viewer, which correlates to how much screen space the hair covers;
- **Hair motion** - Measures the velocity of the section of hair to determine how much simulation accuracy will be needed to model the hair;

The rest of this section will explain these criteria in more detail including why they are important to the viewer, how they are measured, and then how they work together to choose the final hair representation.

3.3.1 Visibility

If a viewer cannot see a section of hair, that section does not need to be simulated or rendered at its highest resolution. The viewer cannot see hair if it is not in the field of view of the camera or if it is completely occluded by the head or other objects in the scene.

If a section of hair in strand representation is normally simulated using s number of skeletons but is occluded by other objects, that section of hair is simulated using one larger strip, and therefore, one skeleton. When that section of hair comes back into view, it is important that the placement and action of the hair are consistent with the case when no levels-of-detail are used at all; therefore, it continues to be simulated. In addition, when a hair section is occluded, it does not need to be rendered at all. Therefore, when a section of hair is occluded, the hair that might normally be represented as either clusters or strands is simulated as strips using fewer skeletons and these sections are not rendered.

3.3.2 Viewing Distance

Hair that is far from the viewer cannot be seen in great detail. The amount of detail that will be seen by the viewer can be estimated by computing the screen space area that the hair covers. As the distance from the viewer to the hair increases, the amount of pixels covered by the hair gets smaller and less detail is viewable. The amount of pixels covered by the hair is calculated to choose the appropriate LOD.

Each level in a hair hierarchy is designed to cover a similar amount of world space, thus the root strip can be used as an estimate to the amount of screen space area a given hair section occupies.

By calculating the amount of pixel coverage the hair will have at its current distance, an appropriate LOD can be chosen. The number of pixels of error for the system is projected into world space to calculate the world space error at the hair's location; this conversion is based on the distance of the hair from the viewer using the following equations:

$$dPP = 0.5 * max(\frac{wR - wL}{W}, \frac{wT - wB}{H})$$

$$WSE = \frac{d * allowedPixelsOfError * dPP}{Near}$$

Here, wR, wL, wT, and wB are the right, left, top, and bottom coordinates of the viewing plane, respectively, and W and H are the width and height of the of the viewing window in pixels. $Near$ is the distance to the near plane, and d is the distance from the camera to the hair section that is currently being tested. The value dPP is the distance per pixel, or amount of object space that a single pixel represents. It is calculated based on the setup of the viewing camera.

The world space error calculated, WSE, is then tested against the error values that have been assigned to each LOD. A representation is chosen by finding the LOD with the maximum error that is still less than the allowable world space error amount. The pre-determined maximum allowable error for each LOD is decided experimentally based on the viewer's preference; it can be easily altered by the viewer in a linear fashion.

3.3.3 Hair Motion

If the hair is not moving at all, then a large amount of computation is not needed to animate it and a lower level-of-detail can be used. When the avatar makes sudden movements, e.g. shaking his or her head, or a large gust of wind blows through the hair, a higher-detailed simulation is used. When a large force is applied to the hair, such as wind, often individual strands can be seen even by a person who is normally too far away to see the individual strands of hair that are not in motion.

A particular LOD is chosen based on hair motion by first determining the skeleton node in the current representation that has the largest velocity. This value is compared to certain thresholds defined for each level of the hierarchy. If the force acting on the skeleton is not high enough to be represented as either strands or clusters, then the hair can be modeled as a strip. The threshold values are based on the thickness of each LOD group. The thicker the group the more easily it should break into smaller groups.

3.3.4 Combining Criteria

At any given time during a simulation, a head of hair is represented by multiple LODs. Each section of hair uses its own parameter values to trigger a transition. The sections of hair that have a root location at the top of the head, and therefore typically more viewable, remain at the strands level longer than the sections of hair that are located at the base of the neck. Thus, even if these two sections are at the same distance from the camera and have the same motion, it is more important that the top layer be represented in more detail since it is in direct view. When determining an appropriate LOD to use, a section of hair is first tested for occlusion. If the hair is not visible to the viewer then it is automatically simulated as a strip and is not rendered. In this case, no other transition tests are needed. If the section of hair is visible, we perform the motion and distance tests described above. The LOD representation is chosen based on whichever of these two tests requires higher detail. The use of different representations for the hair is virtually unnoticeable to the viewer.

3.4 Level-of-Detail Transitions

The hair hierarchy allows the simulation to choose the appropriate discrete representation and resolution for the hair dynamically. The hierarchy is simply traversed selecting the desired hair assemblage. As the simulation moves to a different level in the hair hierarchy either a hair group is divided into multiple groups or several groups are combined into one larger group of hair. The base skeleton makes these transitions smooth and straightforward. Because each hair representation uses the same underlying skeleton for positioning and dynamics, the transitioning algorithm is generalized so that it can be applied at any location in the hierarchy.

A transition is identified following the criteria explained in the previous section. When these tests determine a transition is to occur, the hierarchy either performs adaptive subdivision or adaptive merging of the appropriate hair groups.

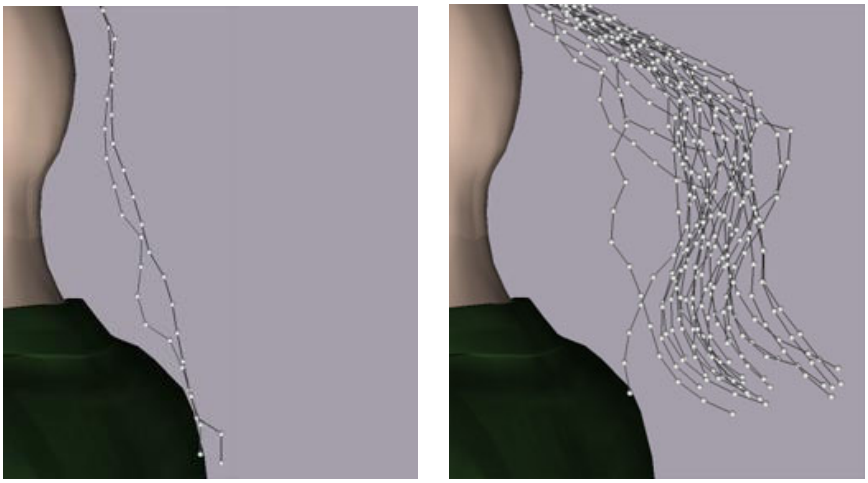

Figure 3.4: Adaptive Subdivision: *Two skeletons (left) are dynamically subdivided into multiple (right).*

3.4.1 Adaptive Subdivision

Using the pre-computed hierarchy, a group of hair can be divided into multiple groups by moving a level down the hierarchy. This becomes a simple process through the use of the base skeleton. Each hair group's skeleton has the same number of control points as its parent skeleton. Furthermore, all of the style properties are the same from parent to child. Accordingly, when a transition to a hair group's children occurs, the child skeletons inherit the dynamic state of their parent skeleton. Each control point in a child skeleton corresponds to a control point in its parent skeleton. When the child groups are created from the parent group, the offset of each child from the parent is stored. When the parent transitions into its children these offsets are used to position the children accordingly.

Figure 3.4 shows two skeletons dynamically subdivide into multiple skeletons as a gust of wind blows through the hair.

3.4.2 Adaptive Merging

Merging multiple child skeletons back into their parent skeleton is, again, rather straightforward. The dynamic states of the children are averaged, including position and velocity values, and the average is then assigned to the parent skeleton.

In order to alleviate visual artifacts that can appear by merging children into a parent skeleton, a transition may only occur if all of the children are ready to transition back into the parent. Furthermore, when merging multiple groups of hair, it is important to avoid a sudden jump in the position of the hair; thus, a positional constraint is imposed on the children for the transition, illustrated in Figure 3.5. First, after the control point positions in the child skeletons are averaged, the distance of the child control points from their corresponding parent control point is calculated (see Figure 3.5b). If this distance for any control point is greater than a certain threshold, the transition will not occur.

It is advantageous to merge groups of hair when possible since it helps to alleviate excess computations. Therefore, if skeletons are near each other but not close enough to merge, the skeletons are subtly pulled closer together so the transition can eventually take place. In this case, control points that fall outside of the first distance threshold are tested against a second, slightly larger, threshold (see Figure 3.5c). If the control points fall within the second threshold, a spring force is used to subtly pull the children into place so a smooth transition may occur (see Figure 3.5d).

3.5 Interactive Dynamic Simulation

In this section, we discuss additional simulation acceleration techniques for hair including an implicit integration scheme, collision detection, and a simulation localization technique based on spatial decomposition that is used to rapidly locate the areas of highest activity. These areas, defined based on various conditions (such as on the user's interaction with the hair during interactive hairstyling), are subsequently simulated with high detail while the simulation resolution of the remaining hair sections is significantly reduced. This process accelerates the dynamic simulation of hair by allocating the majority of the computational resources towards areas of highest importance to the simulation. Simulation and rendering levels can then be achieved that are fast enough to allow a user to actually interact with dynamic hair.

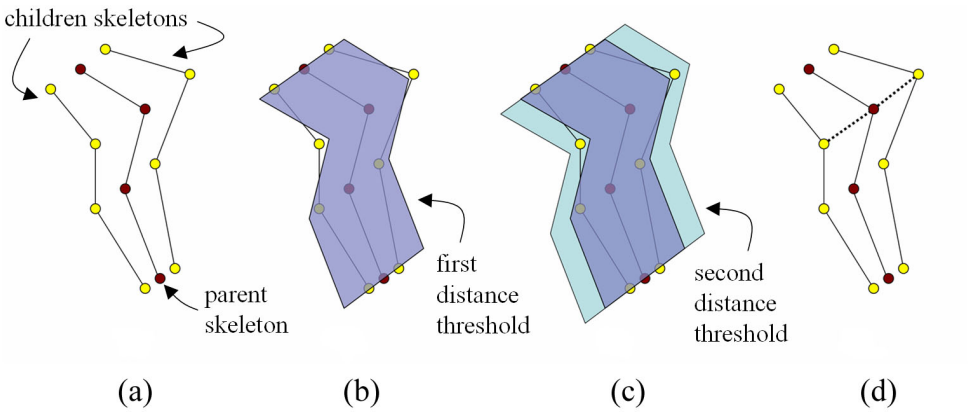

Figure 3.5: Adaptive Merging. *Positional constraints placed on child skeletons merging into parent (a) Parent skeleton (in red) potential position determined by averaging position of child skeletons (in yellow). (b) Distance of child nodes measured from parent node and compared against distance threshold (in blue). (c) Two nodes have greater distance than first threshold, tested against second distance threshold. (d) Nodes are within second threshold, spring force placed between nodes and potential parent position to pull them into place.*

3.5.1 Implicit Integration

Although explicit methods such as Euler or fourth-order Runge-Kutter can be used for this integration, an implicit integration provides greater stability for the simulation. Moreover, many hairstyles, or hair types, require stiff angular springs with high spring constants, for example due to the application of hairspray. Explicit integration schemes are inherently poor for such systems because a very low time step is necessary to avoid instability. The development of this implicit integration scheme not only offers greater stability, but also provides a generality to modeling more diverse hairstyles over the aforementioned explicit techniques. This approach is similar to cloth simulations that use implicit integration for greater stability [BW98a]. This implicit derivation for hair modeling was first presented in Ward and Lin [WL03].

Starting from the basic dynamics model for simulating hair that was first proposed by [AUK92b, KAT93], we use the torque equations due to spring forces calculated by:

$$M_{\theta i} = -k_\theta(\theta_i - \theta_{i0}) \tag{3.1}$$

$$M_{\phi i} = -k_\phi(\phi_i - \phi_{i0}), \tag{3.2}$$

where k_θ and k_ϕ are the spring constants for θ and ϕ, respectively. Furthermore, θ_{i0} and ϕ_{i0} are the specified rest angles and θ_i and ϕ_i are the current angle values.

We will first show how the implicit scheme is derived for the θ-component. Because the bending motion is measured in polar coordinates, the equations will display angular positions, θ and ϕ, angular velocities, ω_θ and ω_ϕ, and angular accelerations, α_θ and α_ϕ.

Rewriting Equation 3.1 as a second-order differential equation returns:

$$\ddot{\theta}(t) = f(\theta(t), \ddot{\theta}(t)) = -k_\theta(\theta_i - \theta_{i0}). \tag{3.3}$$

This can be rewritten as a first-order differential equation by substituting the variables $\alpha_\theta = \ddot{\theta}$ and $\omega_\theta = \dot{\theta}$. The resulting set of first-order differential equations is:

$$\begin{pmatrix} \omega_\theta \\ \alpha_\theta \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} \theta \\ \omega_\theta \end{pmatrix} = \begin{pmatrix} \omega_\theta \\ f(\theta, \omega_\theta) \end{pmatrix}. \tag{3.4}$$

The following formulations for $\triangle\theta$ and $\triangle\omega_\theta$ are derived when using the explicit forward Euler method, where $\triangle\theta = \theta(t_0 + h)$ - $\theta(t_0)$ and $\triangle\omega_\theta = \omega_\theta(t_0 + h)$ - $\omega_\theta(t_0)$ and h is the time step value:

$$\begin{pmatrix} \triangle\theta \\ \triangle\omega_\theta \end{pmatrix} = h\begin{pmatrix} \omega_{\theta 0} \\ -k_\theta(\theta - \theta_0) \end{pmatrix}. \tag{3.5}$$

Instead, an implicit step is used, which is often thought of as taking a backwards Euler step since $f(\theta, \omega_\theta)$ is evaluated at the point being aimed for rather than at the point it was just at. In this case, the set of differential equations changes to the form:

$$\begin{pmatrix} \triangle\theta \\ \triangle\omega_\theta \end{pmatrix} = h\begin{pmatrix} \omega_{\theta 0} + \triangle\omega_\theta \\ f(\theta_0 + \triangle\theta, \omega_{\theta 0} + \triangle\omega_\theta) \end{pmatrix}. \tag{3.6}$$

A Taylor series expansion is applied to f to obtain the first-order approximation:

$$f(\theta_0 + \triangle\theta, \omega_{\theta 0} + \triangle\omega_\theta) \approx f_0 + \frac{\partial f}{\partial \theta}\triangle\theta + \frac{\partial f}{\partial \omega_\theta}\triangle\omega_\theta$$
$$\approx -k_\theta(\theta - \theta_0) - k_\theta\triangle\theta + 0(\triangle\omega_\theta) \approx -k_\theta(\theta - \theta_0) - k_\theta\triangle\theta \tag{3.7}$$

Substituting the approximation of f back into the differential equation of Equation 3.6 yields:

$$\begin{pmatrix} \triangle\theta \\ \triangle\omega_\theta \end{pmatrix} = h \begin{pmatrix} \omega_{\theta 0} + \triangle\omega_\theta \\ -k_\theta(\theta - \theta_0) - k_\theta\triangle\theta \end{pmatrix}. \tag{3.8}$$

Focusing on the angular velocity $\triangle\omega_\theta$ alone and substituting $\triangle\theta = h(\omega_{\theta 0} + \triangle\omega_\theta)$ delivers:

$$\triangle\omega_\theta = h(-k_\theta(\theta - \theta_0) - k_\theta h(\omega_{\theta 0} + \triangle\omega_\theta))$$

Rearranging this equation gives:

$$(1 + k_\theta h^2)\triangle\omega_\theta = -hk_\theta(\theta - \theta_0) - k_\theta h^2\omega_{\theta 0}$$
$$\triangle\omega_\theta = \frac{-hk_\theta(\theta - \theta_0) - h^2 k_\theta \omega_{\theta 0}}{1 + h^2 k_\theta}. \tag{3.9}$$

The change in angular velocity for the θ-component of a skeleton node point, $\triangle\omega_\theta$, is thus given in Equation 3.9, where h is the time step, and $\omega_{\theta 0} = \omega_\theta(t_0)$ is the angular velocity at time t_0. Once $\triangle\omega_\theta$ has been calculated, the change in angular position, $\triangle\theta$, can be calculated from $\triangle\theta = h(\omega_{\theta 0} + \triangle\omega_\theta)$. The same process is applied to the ϕ-component of the angular position and angular velocity for each control point of a skeleton.

Implicit integration allows the use of stiffer springs when warranted, for example, when simulating the bristles of a brush which have different spring constants than the hair on a human head. Using stiff springs with explicit integration on the other hand, requires much smaller time steps to ensure a stable simulation.

3.5.2 Collision Detection and Response

Collision detection and response is typically the most time consuming process for the overall simulation; it can constitute up to 90% of the total animation time. Its intrinsic ability to accelerate collision detection is one of the most appealing contributions of the level-of-detail hair modeling framework. Using a lower level-of-detail to model a section of hair entails using fewer and larger geometric objects, e.g. a single strip versus multiple strands. It is computationally less expensive to check for and handle collisions between a few large objects in comparison to many smaller ones. The LOD system provides an automatic method for using lower LODs whenever possible, thereby accelerating collision detection among other features. Furthermore, the algorithms developed for computing collisions are especially designed for the LOD hair representations giving an accurate and efficient overall collision detection method.

In the rest of this section, we will describe the novel selection of appropriate bounding volumes for each LOD representation. Then, we will explain the process for detecting collisions for both hair-object and hair-hair interactions, including the collision response methods for each type of interaction.

Swept Sphere Volumes

Many techniques have been introduced for collision detection. Common practices have used bounding volumes (BVs) as a method to encapsulate a complex object within a simpler approximation of said object.

We have chosen to utilize the family of "swept sphere volumes" (SSVs) [LGLM00] to surround the hair. SSVs comprise a family of bounding volumes defined by a core skeleton grown outward by some offset. The set of core skeletons may include a point, line, or ngon. Figure 3.6 shows examples of some SSVs, namely a point swept sphere (PSS), a line swept sphere (LSS), and a rectangular swept sphere (RSS). To calculate an SSV, let C denote the core skeleton and S be a sphere of radius r, the resulting SSV is defined as:

$$B = C \oplus S = \{c + r |\ c \in C, r \in S\}. \tag{3.10}$$

To detect an intersection between a pair of arbitrary SSVs a distance test is per-

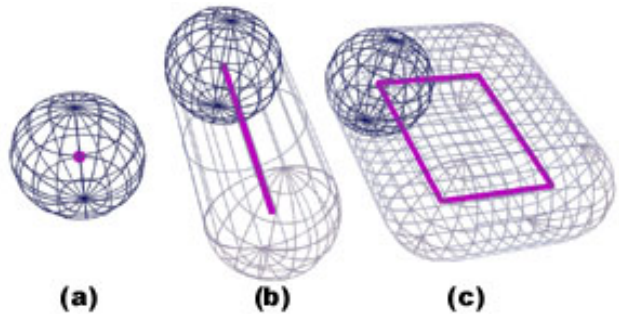

Figure 3.6: Family of Swept Sphere Volumes. *(a) Point swept sphere (PSS); (b) Line swept sphere (LSS); (c) Rectangle swept sphere (RSS). The core skeleton is shown as a bold line or point.*

formed between their corresponding core skeletons and then the appropriate offsets, i.e. the radius of each SSV, are subtracted.

Swept Sphere Volumes for Hair

We have chosen to use the family of SSVs to encapsulate the hair because the shape of the SSVs closely matches the geometry of the hair representations. The SSVs that correspond to the three geometric representations for hair are line swept spheres (LSSs) for the strands and cluster levels, and rectangular swept spheres (RSSs) for the strip level. These SSVs can be used in combination to detect collisions between different representations of hair.

For each rigid segment of the skeleton model, that is, each line segment between two nodes, an SSV bounding volume is pre-computed. For a skeleton with n nodes, there are $n-1$ segments, and thus $n-1$ single SSVs. The variable thickness of each segment defines the radius of the SSV along its length.

In order to compute a BV for a strip, the four control points of the strip that outline a skeletal segment define the area for a RSS to enclose. This is performed for each of the $n-1$ segments along the skeleton. The geometry of the strip is different from the other two representations in that the strip is a surface while the clusters and a collection of strands are volumes. In order to allow the transition from a strip into multiple clusters remain faithful to the volume of hair being depicted an RSS is created for a strip section by surrounding each strip section with a box of certain thickness. Each strip is given a thickness equal to that of its cluster

and strand grouping counterparts. While the strip is rendered as a surface, it acts physically as a volume. Thus, when a transition from a strip into clusters occurs, the *volume* of hair being represented remains constant throughout this process.

For the cluster representation, an LSS is created around the $2m$ control points that define a segment (m control points, as defined in Section 3.1.3, from the cross-section at the top of the segment and m control points at the bottom of the segment). The line segment between the two skeleton control points of each section is used as the core line segment of the line swept sphere.

For individual strands, collision detection is performed for each strand or group of strands, depending on implementation, in a manner similar to that of the clusters. An LSS is computed around the skeleton that defines each segment with a radius defining the thickness. The radius of each LSS is varied based on the thickness of the group of strands.

Hair-Hair Interactions

Because hair is in constant contact with surrounding hair, interactions among hair are important to capture. Ignoring this effect can cause visual disturbances since the hair will not look as voluminous as it should and observing hair passing straight through other hairs creates a visual disruption to the simulation. The typical human head has thousands of hairs. Consequently, testing the $n-1$ sections of each hair group against the remaining sections of hair would be too overwhelming for the simulation even using wisp or LOD techniques. Instead, spatial decomposition is used to create a three-dimensional grid around the area containing the hair and avatar. The average length of the rigid line segments of the skeletons is used as the height, width, and depth of each grid cell. Every time a section of hair moves or the skeleton for simulation is updated, its line swept spheres (LSSs) or rectangular swept spheres (RSSs) are inserted into the grid. An SSV is inserted into the grid by determining which cells first contain the core shape of the SSV (line or rectangle), then the offset of the SSVs are used to determine the remaining inhabited cells. Subsequently, collisions only need to be tested against SSVs that fall within the same cell, refining the collision tests to SSVs with the highest potential for collision.

It is possible for a single SSV to fall into multiple cells. As a result, two separate SSVs can overlap each other in multiple grid cells. To prevent calculating a col-

lision response more than once for the same pair of SSVs, each SSV keeps track of the other SSVs it has encountered in a given time step. Multiple encounters of the same pair of SSVs are ignored.

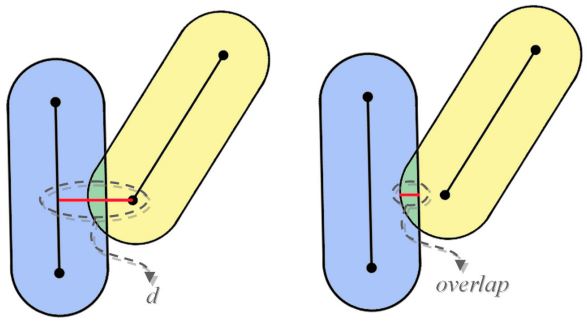

Figure 3.7: Overlap of two line swept spheres (LSSs). *(left) Compute distance d between core lines (right) Subtract offsets to determine overlap value.*

For each pair of SSVs that falls into the same grid cell the distance between their corresponding core skeletons, $s1$ and $s2$, are determined. This distance, d, is subtracted from the sum of the radii of the two SSVs, $r1$ and $r2$, to determine if there is an intersection. Let

$$overlap = d - (r1 + r2) \tag{3.11}$$

If *overlap* is positive then the sections of hair do not overlap and no response is calculated. Figure 3.7 shows the calculation of the overlap of two LSSs. If there is an intersection, their corresponding velocities are set to the average of their initial velocities. This minimizes penetration in subsequent time steps because the sections of hair will start to move together.

Next, following the formulation proposed by [PCP01a], the orientations of the two hair sections will determine how the collision response is handled. The cross product between the core skeletons, $s1$ and $s2$, is computed to determine the orientation of the skeletons in relation to each other. If $s1$ and $s2$ are near parallel, the velocity averaging will be enough to combat their collision, similar to [PCP01a]. Whereas [PCP01a] solely relies on modifying velocities in different manners based on the orientation of the hair sections, using the SSVs to compute collisions makes it straightforward to determine the amount of penetration between corresponding

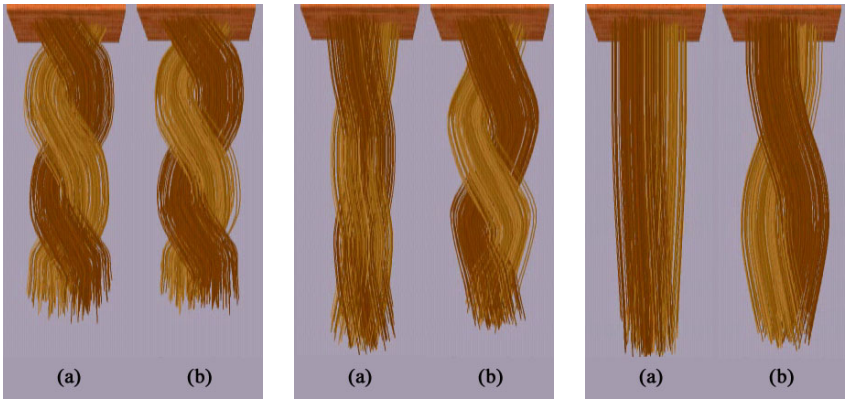

Figure 3.8: Effects of Hair-Hair Collision Detection. *Side-by-side comparison (a) without and (b) with hair-hair collision detection in a sequence of simulation snapshots.*

hair sections. As a result, intersecting hair sections that are not of similar orientations are pushed apart based on their amount of overlap. The extra force exerted to remove hair penetrations help this system to capture finer collision detail than other systems, including intricate braiding or twisting details. The direction to move each hair section is determined by calculating a vector from the closest point on $s1$ to the closest point on $s2$. Each section is moved by half the overlap value and in opposite directions along the vector from $s1$ to $s2$. Figure 3.8 shows the effects of hair-hair interactions in comparison to no hair-hair interactions.

Hair-Object Interactions

Hair can interact with any object in the scene, such as the head or body of the character, where the object is a solid body that allows no penetration. Throughout the rest of this section we will use the terms head and object interchangeably since the collision detection algorithm used for hair-head interactions is applicable to all hair-object interactions.

The spatial decomposition scheme that is used for finding hair-hair interactions is also used to determine potential collisions between the hair and objects in the scene. Therefore, both the hair and the objects must be represented in the grid. The polygons of the avatar, or other objects, are placed into the grid to determine potential collisions with the hair. Object positions only need to be updated within the grid if the object is moving otherwise the initial insertion is sufficient. Grid-

cells that contain both impenetrable triangles and hair geometry are marked to be checked for hair-object collision; only these cells contain a potentially colliding pair. A collision is checked by calculating the distance between the SSV core shape and the triangles and then subtracting the offset of the SSV.

If a section of hair is colliding with the object, the position of the hair section is adjusted so that it is outside of the object. The amount by which to push the hair section is determined by calculating the amount of penetration of the hair section into the object. Then the skeleton is pushed in the direction normal to the object in the amount of the penetration. The section of hair is now no longer colliding with the object. In addition, the velocity of the section of hair is set to zero in the direction towards the object (opposite the direction of the normal), so that the hair is restricted to only move tangential to and away from, the object.

In the next time step, the hair will still be in close proximity to the object. If there is no intersection between the object and the hair it is determined whether the hair is still within a certain distance threshold. If it is within this threshold, then the hair is still restricted so that its velocity in the direction of the object is zero. If it is not within this threshold, then the hair can move about freely.

When hair interacts with an object, a frictional force must be applied. The friction force is calculated by projecting the acceleration of the hair from force onto the plane tangential to the object at the point of contact. The result is the acceleration component that is tangent to the object. The friction force is applied in the opposite direction to oppose the motion of the hair. The magnitude of this force is based on the acceleration of the hair and the frictional coefficient, μ_f, which is dependent upon the surface of the object, where $0 < \mu_f < 1$. The resulting friction force, F_f, becomes:

$$F_f = -\mu_f(F - (F \cdot N)N) \tag{3.12}$$

where F is the force on the hair and N is the normal direction.

3.5.3 Simulation Localization

Interactive hair simulation and rendering is necessary for many applications, including virtual hairstyling tools. An intuitive virtual hairstyling tool needs to take into account user interaction with dynamic hair. Until recently, the complexity

of animating and rendering hair had been too computationally costly to accurately model hair's essential features at desired rates. As a result, many hairstyling methods ignore dynamic simulation and/or user interaction, which creates an unnatural styling process in comparison to what would be expected in practice. In this section, we discuss a simulation localization technique that was originally introduced by Ward *et al.* [WGL06, WGL07] for the creation of an interactive virtual hair salon system. This interactive styling system supports user interaction with dynamic hair through several common hair salon applications, such as applying water and styling products [WGL04].

Spatial decomposition is used to rapidly determine the high activity areas of the hair; these areas are then simulated with finer detail. A uniform grid consisting of axis-aligned cells that encompass the area around the hair and human avatar is employed. This spatial decomposition scheme was previously described for hair-hair and hair-object collision detection. Here, this process is extended to all features of hair simulation, not just collision detection.

Insertion into the Grid

The polygons of the avatar, or other objects, are placed into the grid to determine potential collisions with the hair. Object positions only need to be updated within the grid if the object is moving otherwise the initial insertion is sufficient. The hair is represented in the grid by inserting each SSV of the hair; every time a section of hair moves, or the skeleton for simulation is updated, its line swept spheres (LSSs) or rectangular swept spheres (RSSs) positions are updated in the grid. An SSV is inserted into the grid by determining which cells first contain the core shape of the SSV (line or rectangle), then the offset of the SSVs are used to determine the remaining inhabited cells. Figure 3.9(a) shows the grid cells that contain hair geometry.

When dealing with user interaction with virtual hair, as the user employs an application (e.g. spraying water, grabbing the hair) the grid is used to indicate which portions of the hair are potentially affected by the user's action. As the user moves his or her attention, such as through the use of a PHANToM stylus, its position and orientation are updated. Each application has an *area of influence* that defines where in space its action will have an effect. This area is defined as a triangle for the cutting tool and a cone for the remaining tools. The cone of influence is defined by the application's position, orientation (or direction pointed), length (how

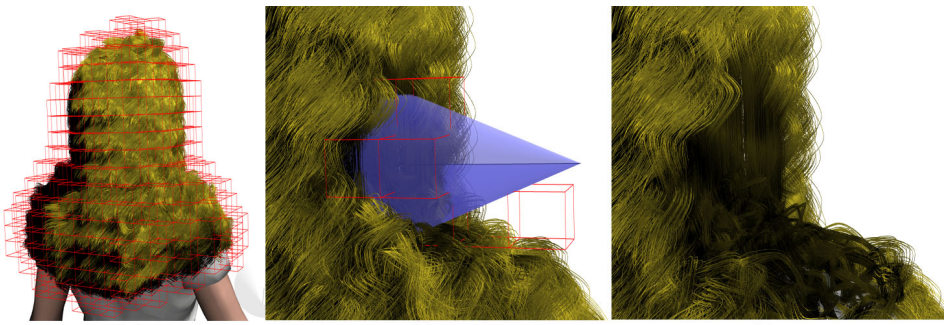

Figure 3.9: (a) Shows all of the grid cells that contain hair geometry (b) Highlights the cells that will be effected by the current application (applying water). (c) Water is applied to some hair, grid allows us to localize each application

far it can reach), and cutoff angle (determining its radius along its length). These properties define the cone's position in the grid. Inserting the cone becomes similar to inserting an LSS, but the offset becomes a variable of distance along the core line (an SSV has a constant offset along its core shape). The triangle for cutting is defined by the space between the open blades of a pair of scissors.

Retrieval from the Grid

Once information has been inserted or updated in the grid, it is retrieved to determine where to check for potential collisions and user interaction. To locate user interactions, the grid maintains a list of grid-cells where the user interaction cone or triangle has been inserted. Any of these grid cells that contain hair geometry are returned and the sections of hair within the cell are independently checked to see if they fall within the area of influence, see Figure 3.9. Using the grid, much fewer sections of hair have to be checked than without it, but the exact hair positions are still checked against the cone or triangle to maintain accuracy.

Multi-Resolution Simulation with the Grid

The grid aids the system to localize the simulation towards the areas of highest importance to the model. Following the criteria discussed earlier, a section of hair's significance is measured by its visibility, motion and viewing distance.

These factors are used to choose the resolution and representation of a section of hair via the hair hierarchy. The simulation localization technique expands upon the motion criterion and adds the user's interaction with the hair to further refine the simulation.

The motion of a section of hair is highly pertinent to the amount of detail needed to simulate it. In the case of interactive styling, most applications performed on hair are localized to a small portion of the hair; the majority of hair thus lies dormant. The sections of hair that are dormant are modeled with a lower LOD representation and resolution, determined by comparison against velocity thresholds as discussed earlier, but here we go a step further by effectively "turning-off" simulation for areas where there is no activity.

Each grid cell keeps track of the activity within the cell, tracking the hair sections that enter and exit the cell. When the action in a given cell has ceased and the hair sections in the cell have a zero velocity, there is no need to compute dynamic simulation due to gravity, spring forces, or collisions. The positions of the hair sections are thus frozen until they are re-activated. The cell is labeled as dormant and does not become active again until either the user interacts with the cell or until a new hair section enters the cell. When a hair section is active, full simulation is performed on it including dynamics of spring forces, gravity, and collision detection and response. Rapid determination of the active cells and hair sections allows the system to allocate the computational resources towards dynamic simulation for the hairs of highest interest to the user.

3.5.4 Results and Discussion

To test the interactive dynamic simulation process described in this section, a virtual hair salon system was implemented, which allows a user to create a hairstyle by directly manipulating dynamic virtual hair. The system allows for a user to dynamically alter the properties through several common hair salon applications (such as cutting, wetting, drying). Further implementation details can be found in Ward *et al.* [WGL07].

Figure 3.10 shows a comparison of real hair under the influence of common hair modeling applications with the virtual salon results under the same conditions. Level-of-detail representations coupled with the simulation localization scheme have accelerated the animation of hair so that a user can actually interact with it.

Figure 3.10: Comparison between real (top) and virtual (bottom) use of common hair salon activities *(from top to bottom) (1) normal, dry hair (2) applying water (3) some wet, some dry hair (4) blow-drying hair.*

Dynamic simulation, including implicit integration, LOD selection, hair applications (wetting, cutting, etc.), and collision detection, to create a hairstyle ran at an average of 0.092 seconds per frame. This figure comprised between 37 to 296 skeleton models, determined on-the-fly throughout the simulation, with an average of 20 control points each. At the finest resolution, the model contained 8,128 rendered strands; throughout the simulation the rendering LOD contained between 6K and 1,311K rendered vertices. Lighting and shadow computations on the GPU were performed in 0.058 seconds/frame on average. The user applications performed to create this style included wetting, cutting and blow-drying. The benchmarks were measured on a desktop PC equipped with an Intel® Xeon™ 2.8 Ghz processor with 2.0 GB RAM and an NVIDIA® GeForce™ 6800 graphics card.

Figure 3.11 shows a detailed performance comparison over the course of an entire simulation between wisps (which are used as the baseline of comparison), LODS alone, and our LODs coupled with simulation localization. The performance of the LODs with simulation localization varies over time due to the user performing different applications on the hair. However, it is clear that the LODs with simulation localization are able to outperform wisps alone as well as LODs alone.

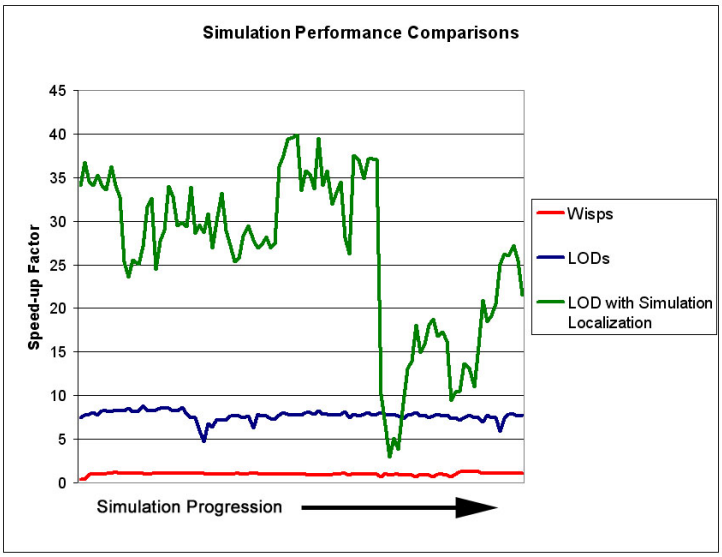

Figure 3.11: Simulation Performance Comparison. *Shows the factor of speed-up for LODs with simulati on localization and LODs over wisps alone. Here, the average runtime of the wisps is used as the baseline for comparison (value of 1 on this chart). Over the course of this simulation, the camera remained at a consistent distance from the figure and the viewer primarily faced the back of the avatar - making distance and occlusion tests have a small overall impact on the LOD choice. Note the LODs with simulation localization overall outperforms both wisps and LODs alone, though the simulation varies over time as the user employs different applications.*

3.6 Conclusion

In this chapter, we have presented the basic features of level-of-detail hair modeling and simulation. These techniques can dynamically change a hair model, allowing a simulation to balance between the visual fidelity and performance speed of the animated hair. To illustrate the effectiveness of the level-of-detail framework for modeling hair, an interactive system for styling hair was implemented using these techniques. The interactive virtual hair salon provides a user interface that obtains 3D input from the user for direct manipulation of dynamic hair and allocates the majority of computational powers towards the hairs that are most significant towards the application. As a result, the hair is simulated at a speed that actually allows a user to directly interact with it.

Kelly Ward, Ming Lin

Chapter 4

Hair Rendering

Steve Marchner

Realistic rendering of human hair requires the handling of both local and global hair properties. To render a full hairstyle, it is necessary to choose an appropriate global representation for hair. Implicit and explicit representations are presented and discussed in Section 4.1. Local hair properties define the way individual hair fibers are illuminated. Section 4.2 describes the scattering properties of hair and reviews the different models that have been proposed to account for those properties. Global hair properties also include the way hair fibers cast shadows on each other; this issue of self-shadowing, handled in Section 4.3, plays a crucial role in volumetric hair appearance. Rendering hair typically requires time-consuming computations, Section 4.4 reviews various rendering acceleration techniques.

4.1 Representing Hair for Rendering

Choices of hair rendering algorithms largely depend on the underlying representations for modeling hair geometry. For example, explicit models require line or triangle-based renderers, whereas volumetric models need volume renderers, or rendering algorithms that work on implicit geometry.

4.1.1 Explicit Representation

With an explicit representation, one has to draw each hair fiber. A hair fiber is naturally represented with a curved cylinder. The early work by Watanabe and Suenaga [WS92] adopted a trigonal prism representation, where each hair strand is represented as connected prisms with three sides. This method assumes that variation in color along the hair radius can be well approximated by a single color. Others use ribbon-like connected triangle strips to represent hair, where each triangle always faces towards the camera. Ivan Neulander [NdP98] introduced a technique that adaptively tessellates a curved hair geometry into polygons depending on the distance to the camera, curvature of hair geometry, etc. At large distances, a hair strand often resembles many hairs. Kong and Nakajima [KN99] exploited this property to reduce the number of rendered hairs by adaptively creating more hairs at the boundary.

Difficulties arise with explicit rendering of tesselated hair geometry due to the unique nature of hair – a hair strand is extremely thin in diameter (0.1 mm). In a normal viewing condition, the projected thickness of a hair strand is much smaller than the size of a pixel. This property causes severe undersampling problems for rendering algorithms for polygonal geometry. Any point sample-based renderer determines a pixel's color (or depth) by a limited number of discrete samples. Undersampling creates abrupt changes in color or noisy edges around the hair. Increasing the number of samples alleviates the problem, but only at slow convergence rates [Mit96] and consequently at increased rendering costs.

LeBlanc *et al.* [LTT91] addressed this issue by properly blending each hair's color using a pixel blending buffer technique. In this method, each hair strand is drawn as connected lines and the shaded color is blended into a pixel buffer. When using alpha-blending, one should be careful with the drawing order. Kim and Neumann [KN02] also use an approximate visibility ordering method to interactively draw hairs with OpenGL's alpha blending.

4.1.2 Implicit Representation

Volumetric textures (or *texels*) [KK89, Ney98] avoid the aliasing problem with pre-filtered shading functions. The smallest primitive is a volumetric cell that can be easily mip-mapped to be used at multiple scales. The cost of ray traversal

is relatively low for short hairs, but can be high for long hairs. Also when hair animates, such volumes should be updated for every frame, making pre-filtering inefficient.

The rendering method of the cluster hair model [YXYW00] also exploits implicit geometry. Each cluster is first approximated by a polygonal boundary. When a ray hits the polygonal surface, predefined density functions are used to accumulate density. By approximating the high frequency detail with volume density functions, the method produces antialiased images of hair clusters. However, this method does not allow changes in the density functions, making hairs appear as if they always stay together.

4.2 Light Scattering in Hair

The first requirement for any hair rendering system is a model for the scattering of light by individual fibers of hair. This model plays the same role in hair rendering as a surface reflection, or local illumination, model does in conventional surface rendering.

4.2.1 Hair Optical Properties

The composition and microscopic structure of hair are important to its appearance. A hair fiber is composed of three structures: the cortex, which is the core of the fiber and provides its physical strength, the cuticle, a coating of protective scales that completely covers the cortex several layers thick (see Figure 1.1 in chapter 1), and the medulla, a structure of unknown function that sometimes appears near the axis of the fiber. As already mentioned in chapter 1, the cross sectional shape varies from circular to elliptical to irregular [Rob94].

Much is known about the chemistry of hair, but for the purposes of optics it suffices to know that it is composed of amorphous proteins that act as a transparent medium with an index of refraction $\eta = 1.55$ [Rob94, SGF77]. The cortex and medulla contain pigments that absorb light, often in a wavelength-dependent way; these pigments are the cause of the color of hair.

4.2.2 Notation and Radiometry of Fiber Reflection

Our notation for scattering geometry is summarized in Figure 4.1. We refer to the plane perpendicular to the fiber as the *normal plane*. The direction of illumination is ω_i, and the direction in which scattered light is being computed or measured is ω_r; both direction vectors point away from the center. We express ω_i and ω_r in spherical coordinates. The inclinations with respect to the normal plane are denoted θ_i and θ_r (measured so that 0 degree is perpendicular to the hair). The azimuths around the hair are denoted ϕ_i and ϕ_r, and the relative azimuth $\phi_r - \phi_i$, which is sufficient for circular fibers, is denoted $\Delta\phi$.

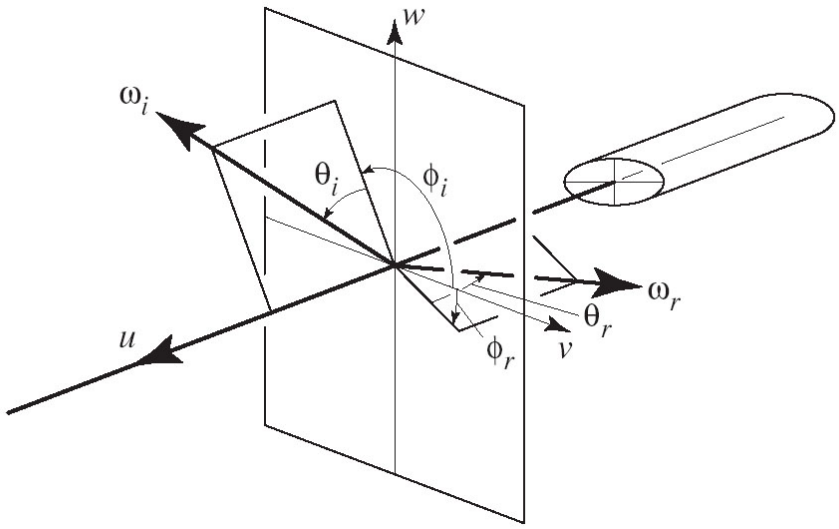

Figure 4.1: Notation for scattering geometry

Because fibers are usually treated as one-dimensional entities, light reflection from fibers needs to be described somewhat differently from the more familiar surface reflection. Light scattering at a surface is conventionally described using the bidirectional reflectance distribution function (BRDF), $f_r(\omega_i, \omega_r)$. The BRDF gives the density with respect to the projected solid angle of the scattered flux that results from a narrow incident beam from the direction ω_i. It is defined as the ratio of surface radiance (intensity per unit projected area) exiting the surface in direction ω_r to surface irradiance (flux per unit area) falling on the surface from a

differential solid angle in the direction ω_i:

$$f_r(\omega_i, \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)}.$$

Under this definition, the radiance due to an incoming radiance distribution $L_i(\omega_i)$ is

$$L_r(\omega_r) = \int_{H^2} f_r(\omega_i, \omega_r) L_i(\omega_i) \cos\theta_i d\omega_i$$

where H^2 is the hemisphere of directions above the surface.

Light scattering from fibers is described similarly, but the units for measuring the incident and reflected light are different because the light is being reflected from a one-dimensional curve [MJC+03b]. If we replace "surface" with "curve" and "area" with "length" in the definition above we obtain a definition of the scattering function f_s for a fiber: "the ratio of *curve radiance* (intensity per unit projected length) exiting the curve in direction ω_r to *curve irradiance* (flux per unit length) falling on the curve from a differential solid angle in the direction ω_i." The curve radiance due to illumination from an incoming radiance distribution L_i is

$$L_r^c(\omega_r) = D \int_{H^2} f_s(\omega_i, \omega_r) L_i(\omega_i) \cos\theta_i d\omega_i$$

where D is the diameter of the hair as seen from the illumination direction.

This transformation motivated Marschner et al. [MJC+03b] to introduce curve radiance and curve irradiance. Curve radiance is in some sense halfway between the concepts of radiance and intensity, and it describes the contribution of a thin fiber to an image independent of its width. Curve irradiance measures the radiant power intercepted per unit length of fiber and therefore increases with the fiber's width. Thus, given two fibers with identical properties but different widths, both will have the same scattering function but the wider fiber will produce a brighter curve in a rendered image because the wider fiber intercepts more incident light. This definition is consistent with the behavior of real fibers: very fine hairs do appear fainter when viewed in isolation.

Most of the hair scattering literature does not discuss radiometry, but the above definitions formalize the common practice, except that the diameter of the hair is normally omitted since it is just a constant factor. The factor of $\cos\theta_i$ is often included in the model, as was common in early presentations of surface shading models.

4.2.3 Reflection and Refraction in Cylinders

For specular reflection, a hair can be modeled, to a first approximation, as a transparent (if lightly pigmented) or purely reflecting (if highly pigmented) dielectric cylinder. The light-scattering properties of cylinders have been extensively studied in order to inversely determine the properties of optical fibers by examining their scattering [ALS98, Mar74, MTM98].

As first presented in graphics by Kajiya and Kay [KK89] (their scattering model is presented in Section 4.2.5), if we consider a bundle of parallel rays that illuminates a smooth cylinder, each ray will reflect across the local surface normal at the point where it strikes the surface. These surface normals are all perpendicular to the fiber axis–they lie in the normal plane. Because the direction of each reflected ray is symmetric to the incident direction across the local normal, all the reflected rays will make the same angle with the normal plane. This means that the reflected distribution from a parallel beam due to specular reflection from the surface lies in a cone at the same inclination as the incident beam.

For hairs that are not darkly pigmented, the component of light that is refracted and enters the interior of the hair is also important. As a consequence of Bravais's Law [Tri70], a corrolary of Snell's Law that is often used to describe refractions through crystals with a cylinder-like structure, the directions of the rays that are refracted through the cylinder surface also fall on a cone centered on the cylinder axis. The same holds for the refractions as the rays exit the cylinder. Therefore all specularly reflected light from a smooth cylinder will emit on the same cone as the surface reflection, no matter what sequence of refractions and internal reflections it may have taken.

4.2.4 Measurements of Hair Scattering

In cosmetics literature, some measurements of incidence-plane scattering from fibers have been published. Stamm et al. [SGF77] made measurements of reflection from an array of parallel fibers. They observed several remarkable departures from the expected reflection into the specular cone: there are two specular peaks, one on either side of the specular direction, and there is a sharp true specular peak that emerges at grazing angles. The authors explained the presence of the two peaks using an incidence-plane analysis of light reflecting from the tilted scales

that cover the fiber, with the surface reflection and the first-order internal reflection explaining the two specular peaks.

A later paper by Bustard and Smith [BS91] reported additional measurements of single fibers, including measuring the four combinations of incident and scattered linear polarization states. They found that one of the specular peaks was mainly depolarized while the other preserved the polarization. This discovery provided additional evidence for the explanation of one lobe from the surface reflection and one from the internal reflection.

Bustard and Smith also discussed preliminary results of an azimuthal measurement, performed with illumination and viewing perpendicular to the fiber. They reported finding bright peaks in the azimuthal distribution and speculated that they were due to caustic formation, but they did not report any data.

Marschner et al. [MJC+03b] reported measurements of single fibers in more general geometries. In addition to incidence plane measurements, they presented normal plane measurements that show in detail the peaks that Bustard and Smith discussed and how they evolve as a strand of hair is rotated around its axis. The authors referred to these peaks as "glints" and showed a simulation of scattering from an elliptical cylinder that predicts the evolution of the glints; this clearly confirmed that the glints are caused by caustic formation in internal reflection paths. They also reported some higher-dimensional measurements that show the evolution of the peaks with the angle of incidence, which showed the full scattered distribution for a particular angle of incidence.

4.2.5 Models for Hair Scattering

The earliest and most widely used model for hair scattering is Kajiya and Kay's model which was developed for rendering fur [KK89]. This model includes a diffuse component and a specular component:

$$S(\theta_i, \phi_i, \theta_r, \phi_r) = k_d + k_s \frac{\cos^p(\theta_r + \theta_i)}{\cos(\theta_i)}.$$

Kajiya and Kay derived the diffuse component by integrating reflected radiance across the width of an opaque, diffuse cylinder. Their specular component is simply motivated by the argument from the preceding section that the ideal specular reflection from the surface will be confined to a cone and therefore the reflection

Figure 4.2: Comparison between Kajiya's model (left), Marschner's model (middle) and real hair (right).

from a non-ideal fiber should be a lobe concentrated near that cone. Note that neither the peak value nor the width of the specular lobe changes with θ or ϕ.

Banks [Ban94] later re-explained the same model based on more minimal geometric arguments. For diffuse reflection, a differential piece of fiber is illuminated by a beam with a cross section proportional to $\cos\theta_i$ and the diffusely reflected power emits uniformly to all directions.[1] For specular reflection, Fermat's principle requires that the projection of the incident and reflected rays onto the fiber be the same.

In another paper on rendering fur, Goldman [Gol97], among a number of other refinements to the aggregate shading model, proposed a refinement to introduce azimuthal dependence into the fiber scattering model. He multiplied both terms of the model by a factor f_{dir} that can be expressed in the current notation as:

$$f_{dir} = 1 + a\cos\Delta\phi.$$

Setting $a > 0$ serves to bias the model toward backward scattering, while setting $a < 0$ biases the model towards forward scattering.[2]

Tae-Yong Kim [Kim02] proposed another model for azimuthal dependence, which accounts for surface reflection and transmission using two cosine lobes. The sur-

[1] Banks does not discuss why uniform curve radiance is the appropriate sense in which the scattered light should be uniform.

[2] In Goldman's original notation $a = (\rho_{reflect} - \rho_{transmit})/(\rho_{reflect} + \rho_{transmit})$. A factor of $\frac{1}{2}(\rho_{reflect} + \rho_{transmit})$ can be absorbed into the diffuse and specular coefficients.

face reflection lobe derives from the assumption of mirror reflection with constant reflectance (that is, ignoring the Fresnel factor), and the transmission lobe is designed empirically to give a forward-focused lobe. The model is built on Kajiya-Kay in the same way Goldman's is, defining:

$$g(\phi) = \begin{cases} \cos\phi & -\frac{\pi}{2} < \phi < \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases}$$

This model is Kajiya and Kay's model multiplied by:

$$f_{dir} = a\, g(\Delta\phi/2) + g(k(\Delta\phi - \pi))$$

where a is used to balance forward and backward scattering and k is a parameter to control how focused the forward scattering is. The first term is for backward (surface) scattering and the second term is for forward (transmitted) scattering.

Marschner *et al.* [MJC+03b] proposed the most complete physically based hair scattering model to date. Their model makes two improvements to Kajiya and Kay's model: it predicts the azimuthal variation in scattered light based on the ray optics of a cylinder, and it accounts for the longitudinal separation of the highlight into surface-reflection, transmission, and internal-reflection components that emerge at different angles. The azimuthal component of the model is based on a ray analysis that accounts for focusing and dispersion of light, absorption in the interior, and Fresnel reflection at each interaction. The longitudinal component models the shifts of the first three orders of reflection empirically using lobes that are displaced from the specular cone by specific angles.

4.2.6 Light Scattering on Wet Hair

The way light scatters on hair is changed when hair becomes wet. Jensen *et al.* [JLD99] noted that when objects become wet they typically appear darker and shinier; hair behaves the same way. Bruderlin [Bru99] and Ward *et al.* [WGL04] altered previous light scattering models to capture the effects of wet fur and wet hair, respectively.

As hair becomes wet, a thin film of water is formed around the fibers, forming a smooth, mirror-like surface on the hair. In contrast to the naturally rough, tiled surface of dry hair, this smoother surface creates a shinier appearance of the hair

due to increased specular reflections. Furthermore, light rays are subject to total internal reflection inside the film of water around the hair strands, contributing to the darker appearance wet hair has over dry hair. Moreover, water is absorbed into the hair fiber, increasing the opacity value of each strand leading to more aggressive self-shadowing (see Section 4.3).

Bruderlin [Bru99] and Ward *et al.* [WGL04] modeled wet strands by increasing the amount of specular reflection. Furthermore, by increasing the opacity value of the hair, the fibers attain a darker and shinier look, resembling the appearance of wet hair.

4.3 Hair Self-Shadowing

Figure 4.3: Importance of self-shadowing on hair appearance. (left) No shadows compared to (right) Shadows computed using a light-oriented map [BMC05].

Hair fibers cast shadows onto each other, as well as receiving and casting shadows from and to other objects in the scene. Self-shadowing creates crucial visual patterns that distinguish one hairstyle from another, see Figure 4.3. Unlike solid objects, a dense volume of hair exhibits complex light propagation patterns. Each hair fiber transmits and scatters rather than fully blocks the incoming lights. The strong forward scattering properties as well as the complex underlying geometry make the shadow computation difficult.

One can ray trace hair geometry to compute shadow, whether hair is represented by implicit models [KK89] or explicit models [MJC+03b]. For complex geometry, the cost of ray traversal can be expensive and many authors turn to caching schemes for efficiency. Two main techniques are generally used to cast self-shadows into volumetric objects: ray casting through volumetric densities and shadow maps.

4.3.1 Ray-casting through a Volumetric Representation

With implicit hair representations, one can directly ray trace volume density [YXYW00], or use two-pass shadowing schemes for volume density [KK89]; the first pass fills volume density with shadow information and the second pass renders the volume density.

4.3.2 Shadow Maps

LeBlanc [LTT91] introduced the use of the shadow map, a depth image of hair rendered from the light's point of view. In this technique, hair and other objects are rendered from the light's point of view and the depth values are stored. Each point to be shadowed is projected onto the light's camera and the point's depth is checked against the depth in the shadow map. Kong and Nakijima [KN99] extended the principle of shadow caching to the visible volume buffer, where shadow information is stored in a 3D grid.

In complex hair volumes, depths can vary radically over small changes in image space. The discrete nature of depth sampling limits shadow buffers in handling hair. Moreover, lights tend to gradually attenuate through hair fibers due to forward scattering. The binary decision in depth testing inherently precludes such light transmission phenomena. Thus, shadow buffers are unsuitable for volumetric hair.

The transmittance $\tau(p)$ of a light to a point p can be written as:

$$\tau(p) = \exp(-\Omega), \text{ where } \Omega = \int_0^l \sigma_t(l')dl'.$$

l is the length of a path from the light to p, σ_t is the extinction (or density) function along the path. Ω is the opacity thickness (or accumulated extinction function).

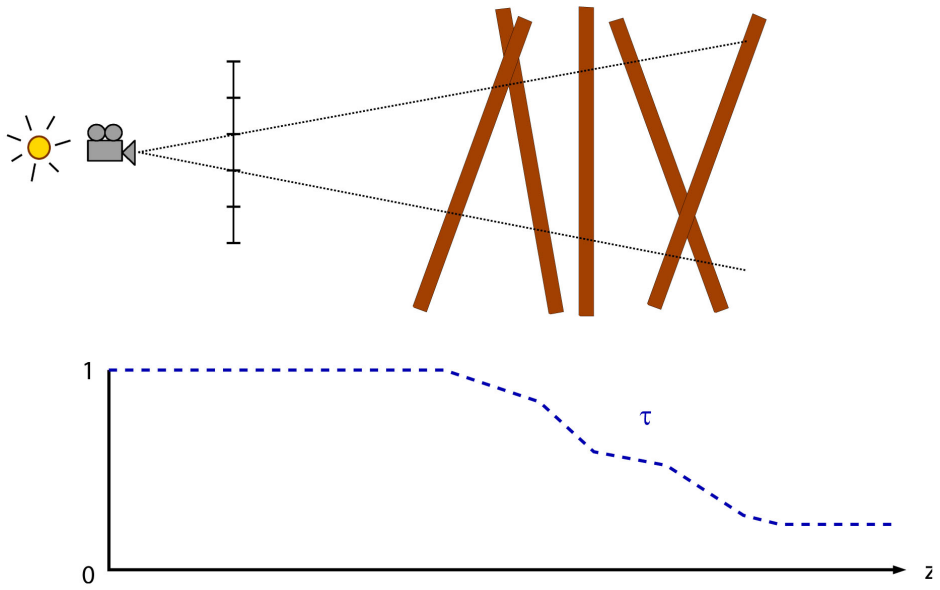

Figure 4.4: Top: a beam of light starting at the shadow camera origin (*i.e.*, the light source) and passing through a single pixel of the deep shadow map. Bottom: the corresponding transmittance (or visibility) function τ, stored as a piecewise linear function.

In the deep shadow maps technique [LV00], each pixel stores a piecewise linear approximation of the transmittance function instead of a single depth, yielding more precise shadow computations than shadow maps, see Figure 4.4 for an illustration. The transmittance function accounts for two important properties of hair.

Fractional Visibility: In the context of hair rendering, the transmittance function can be regarded as a fractional visibility function from the light's point of view. If more hair fibers are seen along the path from the light, the light gets more attenuated (occluded), resulting in less illumination (shadow). As noted earlier, visibility can change drastically over the pixel's extent. To handle this partial visibility problem, one should accurately compute the transmission function by correctly integrating and filtering all the contributions from the underlying geometry.

Translucency: A hair fiber not only absorbs, but also scatters and transmits the incoming light. Assuming that the hair fiber transmits the incoming light only in a forward direction, the translucency is also handled by the transmittance function.

Noting that the transmittance function typically varies radically over image space, but gradually along the light direction, one can accurately approximate the transmittance function with a compact representation. Deep shadow maps [LV00] use a compressed piecewise linear function for each pixel, along with special handling for discontinuities in transmittance. Figure 4.3 shows a comparison of hair geometry with and without shadows using the deep shadow maps algorithm.

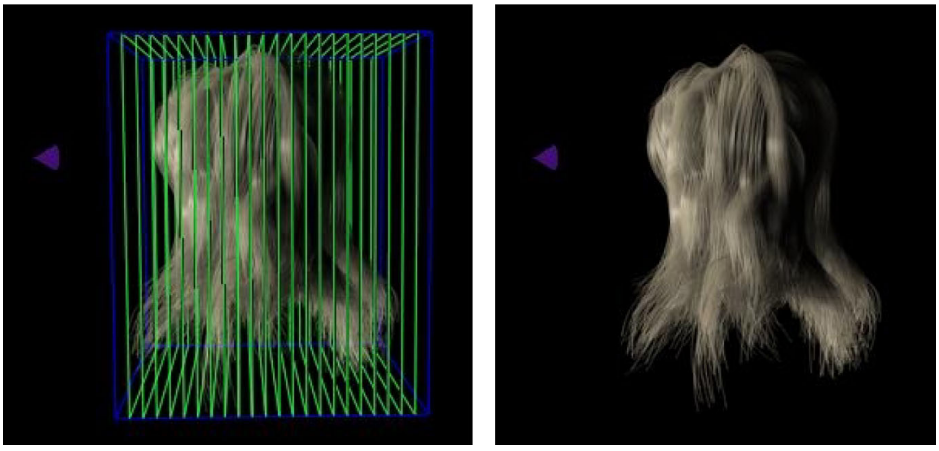

Figure 4.5: Opacity Shadow Maps. Hair volume is uniformly sliced perpendicular to the light direction into a set of planar maps storing alpha values (top). The resulting shadowed hair (bottom).

Opacity shadow maps [KN01] further assume that such transmittance functions always vary smoothly, and can thus be approximated with a set of fixed image caches perpendicular to the lighting direction (see Figure 4.5). By approximating the transmittance function with discrete planar maps, opacity maps can be efficiently generated with graphics hardware (see Section 4.4.3). Linear interpolation from such maps facilitates fast approximation to hair self-shadows.

4.4 Rendering Acceleration Techniques

Accurately rendering complex hairstyles can take several minutes for one frame. Many applications, such as games or virtual reality, require real-time rendering

of hair. These demands have initiated recent work to accelerate precise rendering algorithms by simplifying the geometric representation of hair, by developing fast volumetric rendering, or by utilizing recent advances in graphics hardware.

4.4.1 Approximating Hair Geometry

Section 4.2 explained the structure of hair and showed that hair fibers are actually quite complex. Simplifying this geometry, using fewer vertices and rendering fewer strands, is one strategy for accelerating hair rendering. Removing large portions of hair strands can be distracting and unrealistic, therefore surfaces and strips have been used for approximating large numbers of hair strands [KH00, KH01, GZ02, KHS04].

These two-dimensional representations resemble hair by texture mapping the surfaces with hair images and using alpha mapping to give the illusion of individual hair strands. Curly wisps can be generated by projecting the hair patch onto a cylindrical surface [KHS04].

Level of detail (LOD) representations used by Ward *et al.* [WLL+03, WL03] (see chapter 3) for accelerating the dynamic simulation of hair, also accelerates hair rendering. Using a coarse LOD to model hair that cannot be seen well by the viewer requires rendering fewer vertices with little loss in visual fidelity. As a result, the time required to calculate light scattering and shadowing effects is diminished by an order of magnitude.

4.4.2 Interactive Volumetric Rendering

Bando *et al.* [BCN03] modeled hair as a set of connected particles, where particles represent hair volume density. Their rendering method was inspired by fast cloud rendering techniques [DKY+00] where each particle is rendered by splatting a textured billboard, both for self-shadowing computation and final rendering. This method runs interactively, but it does not cast very accurate shadows inside hair.

Bertails *et al.* [BMC05] use a light-oriented voxel grid to store hair density values, which enables them to efficiently compute accumulative transmittance inside the hair volume. Transmittance values are then filtered and combined with diffuse and specular components to calculate the final color of each hair segment. Though

very simple, this method yields convincing interactive results for animated hair (see Figure 4.6). Moreover, it can easily be parallelized to increase performance.

Figure 4.6: Interactive hair self-shadowing processed by accumulating transmittance values through a light-oriented voxel grid [BMC05]. (left) Animated hair without self-shadows; (right) Animated hair with self-shadows.

4.4.3 Graphics Hardware

Many impressive advances have been made recently in programmable graphics hardware. Graphics processor units (GPUs) now allow programming of more and more complex operations through dedicated languages, such as Cg. For example, various shaders can directly be implemented on the hardware, which greatly improves performance. Currently, the major drawback of advanced GPU programming is that new features are neither easy to implement nor portable across different graphics cards.

Heidrich and Seidel [HS98] efficiently render anisotropic surfaces by using OpenGL texture mapping. Anisotropic reflections of individual hair fibers have also been implemented with this method for straightforward efficiency.

As for hair self-shadowing, some approaches have recently focused on the acceleration of the opacity shadow maps algorithm (presented in Section 4.3.2), by using the recent capabilities of GPUs. Koster *et al.* [KHS04] exploited graphics hardware by storing all the opacity maps in a 3D texture, to have the hair self-shadow computation done purely in graphics hardware. Using textured strips

to simplify hair geometry (as seen in Section 4.4.1), they achieve real-time performance. Mertens *et al.* [MKBR04] explored efficient hair density clustering schemes suited for graphics hardware, achieving interactive rates for high quality shadow generation in dynamically changing hair geometry. Finally, a real-time demonstration showing long hair moving in the sea was presented by NVidia in 2004 [ZFWH04] to illustrate the new capabilities of their latest graphics cards (see http://www.nzone.com/object/nzone_nalu_home.html).

Chapter 5

Hair in Feature Production

Sunil Hadap, Zoran Kačić-Alesić, Tae-Yong Kim

5.1 Strand and Hair Simulation at PDI/DreamWorks - Madagascar and Shrek The Third

In this section, we would like to exemplify the versatile use of the dynamic strand primitive, introduced in section 1.3, for character dynamics and visual effects. The proposed methodology is extensively used in the production of Madagascar and Shrek The Third. With each example, we would like to highlight the aspects of the formulation that is most relevant. The methodology and the tools based on it are continually being effective for simulation of strands and long hair in upcoming productions at DreamWorks Animation such as Bee Movie.

The example of lemurs dancing with fancy flower lanterns in (Figure 5.1), highlights the importance of stable and consistent coordinate frame along the strand. The tip of the strand is made heavy by using the length-wise variation of *mass per unit length* parameter, and the flower geometry is simply parented to the coordinate frame at the tip of the strand. Subtle twist dynamics adds to the realism.

The foliage simulation is a great example of branched dynamics. The individual trees are composed of hierarchy of strands, some of the segments being very stiff towards the trunk. One can follow the physically believable motion of trees under

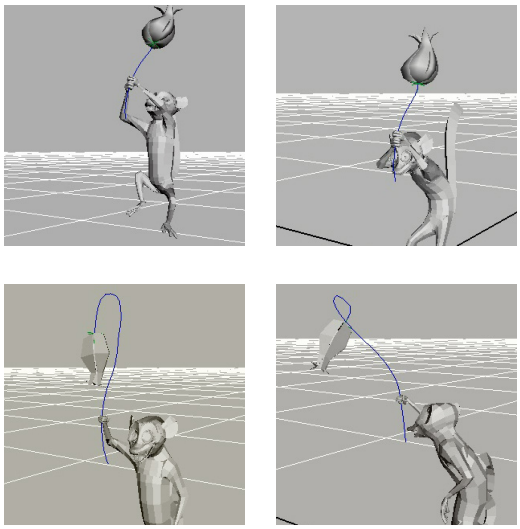

Figure 5.1: Lemurs dancing with the lanterns (Madagascar) - consistent coordinates, twist dynamics

Figure 5.2: Moving Jungle (Madagascar) - branched dynamics, stiff articulation, strong external force fields, scalability

Figure 5.3: Donkey's Ears (Shrek The Third) - posable dynamics

the influence of (strong) external force field such as wake and turbulence. It is also evident that the strand system is very scalable. Each tree typically has 50-100 segments and there are around 1000 trees in the "Blown by Horn" shot (video 3).

Donkey's ear exemplifies posable dynamics. Animators precisely control the subtle secondary dynamic motion around the animated poses, using time-varing *pose strength* parameter.

The bun followed by the long braid is composed of a single strand. The very stiff intial section gives the subtle interesting bounce and twist to the bun. The flexible section corresponds to the braid. The *local dynamics* parameter is used to control the amount of "floppyness" the braid exibits.

The simulation of curly bangs illustrate the ability of the system to handle "stiff" equations arising from the intricate rest shape. The rest shape is adjusted to account for the shape change due to gravity.

The long hair simulations highlight the effectiveness of the collision response model. The accurate computation of velocity and acceleration of the base joint results in highly realistic hair motion, where as *time scale* parameter gives control.

We have not done a comprehensive performance analysis and optimization of the

Figure 5.4: Rapunzel's Braid (Shrek The Third) - "stiff" equations, local dynamics

Figure 5.5: Guinevere's Curly Bangs (Shrek The Third) - intricate and zero-gravity rest shape

Figure 5.6: Sleeping Beauty's Long Hair (Shrek The Third) - accurate base acceleration, elaborate collision response, time scale

Oriented Strands system yet. Nevertheless, we would like to state the typical performance numbers for the hair simulations, as they represent the most of the dynamic complexities. The simulation of curly bangs uses 9 strands having an average 15 segments, each. The simulation runs at interactive rate of 2-4 Hz. The long hair simulations use 190 strands with 20-25 segments each. The simulations take less than 20 seconds per animation frame. The complexity of the strand dynamics is linear time in the total number of segments n. Whereas, the collision response is $O(m^2)$ in m number of collision points. Recently, we tried to analyze the convergence characteristics of the solver. We found that the solver uses sophisticated error control and heuristics, which result in a very wide variation in the number of non-linear iteration the solver takes. For the long hair simulations, the number varies from 2 to 213 iterations, with mean at 18.3 iterations. In the advent of multi-core and multi-cpu workstations, we would like to note that the computations of individual strands are *embarrassingly parallel*.

5.1.1 Conclusion, Limitations and Future Work

The simulation system of Oriented Strands has found widespread applications in feature animation and visual effects. We would like to attribute the successful

usage of Oriented Strands to the robustness coming from the implicit formulation and the comprehensive collision response model, the intuitive workflow coming from local space formulation, physically based stiffness and collision models. In addition, innovative concepts such as time scale, local dynamics, posable dynamics, zero-gravity rest shape make Oriented Strands system "art directable".

In this methodology, our focus has been "stiff" dynamics of serial open-chain multi-body systems with constraints and collision response. Fortunately, the DAE based formulation can be extended to include closed-loops [RJFdJ04]. Unfortunately, the analytical constraints and collision response model discussed so far do not readily fit the framework of closed-loops. Thus, in future we would like to extend, or develop new, methodologies to include closed-loops. Intricate jewelry on animated characters is our main motivation.

Other limitations of the proposed methodology are

- The approach is computationally expensive as compared to previous methods in [Had03, CCK05b]. It would not scale well to do e.g. fur dynamics.
- Although one can incorporate stretch in the strand system by animating lengths of rigid segments, the system does not handle stretch dynamically.
- Developing and implementing constraints and collision response model is not as straightforward as compared to maximal coordinate formulation [CCK05b].

5.1.2 Acknowledgments

I would like to take this opportunity to acknowledge the great teamwork at PDI/DreamWorks. I would like to thank Dave Eberle and Deepak Tolani for their great collaboration during the development of the system, Scott Singer, Arnauld Lamorlette, Scott Peterson, Francois Antoine, Larry Cutler, Lucia Modesto, Terran Boylan, Jeffrey Jay, Daniel Dawson, Lee Graft, Damon Riesberg, David Caeiro Cebrian, Alain De Hoe and everyone who directly influenced the development and tirelessly worked towards the successful use of the system in production, my supervisors Lisa Mary-Lamb, Andrew Pearce and Sanjay Das for supporting me all along, and anyone I forgot to mention.

5.2 Strand and Hair Simulation at ILM

The quest for ever increasing realism in visual effects has reached a point where many viewers, including experts, frequently cannot tell what aspects of a live action movie were created using computer graphics techniques. Often, only the improbability that a scene was shot using practical means provides a clue. It is now commonly expected that hair, skin, muscles, clothing, jewelry, and accessories of digital characters look and move in a way indistinguishable from real ones. And without exceptions, these aspects of digital characters are not supposed to attract attention unless intended so by the movie maker.

Dynamics simulations are a significant contributor to this apparent success. At *Industrial Light & Magic* we use them extensively to achieve believable motion of digital characters. Simulation of hair and strands is an integral part of a collection of techniques that we commonly refer to as *structural dynamics*.

As much as we like to celebrate our achievements in the area of simulation, the other main point that we hope to convey is that we still have a long way to go in this field. In many ways, recent successes have just opened the door to a widespread use of simulations. We want to describe not only what has been done, but also what we would like to do but have not been able, so far. There are many interesting and challenging problems left to be solved. We hope this presentation provides motivation to tackle some of these issues.

5.2.1 Overview

The survey paper [WBK+07] provides a very good overview of many techniques used in our hair pipeline at ILM. Typically, we think of our pipeline as consisting of four distinct stages:

- *Hair placement and styling:* an artist driven, interactive modeling task done in our proprietary 3D application called Zeno. Our general free-form surface sculpting tools were augmented to satisfy hair specific needs such as hair length and rooting preservation, twisting and curling, interpolated placement, hair extension and clipping. These tools are used to create a set of "guide" hairs (B-spline curves) representative of the overall hair style, and are also very useful for touching up simulation results. The number

of guide hairs can vary widely, depending on the complexity and coverage, from several hundred to many thousands.

Although still subject to modifications, this is a mature and well established component of our pipeline, with nearly a decade of production use.

- *Simulation:* interactive tools and scripts for creation and editing of simulation meshes/rigs, tuning simulation controls and parameters, and running simulations in Zeno. This is among the most complex parts of our pipeline and is the main topic of our discussion.
- *Hair generation:* a procedural technique for creating a complete set of hair strands (a full body of hair) from a relatively small number of guide hairs. The number of generated hairs typically ranges from mid tens of thousands to several million. Finer details of hair tufting and wisping, jitter, and irregularities are almost exclusively handled at this level.

  This is the oldest component of our pipeline, having roots in the techniques developed for "Jumanji" in 1995. It still undergoes frequent show and creature specific modifications, and has recently been completely overhauled.
- *Lighting and rendering.* We used to render hair using our own in-house particle and hair renderer called Prender, but for several years now we have relied on RenderMan exclusively to render hairs as B-spline curves. Setting up the lighting is done using our proprietary tools inside Zeno.

5.2.2 Dynamics of Hair and Strands

Animation of hair and strands can be achieved using the same combination of keyframe techniques, inverse kinematics, procedural deformation, and motion capture that is applied to the body and skin of a digital character. As long as hair follows along and remains properly oriented and rooted on the underlying surface, the results may look convincing. This is often sufficient for short hair and fur, and for background characters.

In other cases, particularly when appearance of compliance with the laws of physics is desired, traditional and manual animation techniques fall short or are at best tedious. To a degree this is also true for other aspects of digital character animation, but it is particularly problematic for fast moving long and medium length hair.

Figure 5.7: Wookies. How many are not CG? *Star Wars: Episode III - Revenge of the Sith* (2005)

Physically based simulations have provided a solution that we increasingly depend on at ILM.

We have traditionally relied on our own proprietary simulation tools, although our artists have access to and use when appropriate dynamics tools inside vendor packages such as Maya. Over the years we have made several major revisions to our software - our hair and cloth simulation tools are currently in their third generation, and rigid body and flesh/skin tools are in their second. Over the past several years we have collaborated closely with Ron Fedkiw and his graduate students at Stanford, and have migrated our simulation engines to use PhysBAM, which is also our fluid simulation engine.

With regard to a physical model used as representative of hair dynamics, we have been firmly in the mass-spring camp. This is also true for our cloth, skin, and flesh simulation systems. Rigid body dynamics has also been one of the mainstays of our pipeline - simulations of ropes, chains, and accessories that dangle from digital characters and mechanical structures have long been employed in production. In [OBKA03] we describe how our rigid body dynamics system was used to create extremely complex, convincing character performances for *The Hulk* (2003). And in [KANB03] we describe how fundamentally similar or identical techniques can be used for deformable and rigid body simulations.

Sometimes, it is not obvious what types of simulation would be the most appropriate or the most cost effective for a particular character. For digital doubles and

furry creatures simulation of thin strands of hair is often the obvious choice. But when the hair is braided, as it is for some Wookies (figure 5.7), or made of kelp and other types of seaweed (figure 5.9), as it is for some pirates of the Caribbean, or the strands are actually octopus-like tentacles, as they are on the Davy Jones character (figures 5.8 and 5.17), techniques commonly associated with cloth, flesh, and articulated rigid body simulations become equally or more attractive.

Figure 5.8: Davy Jones

Figure 5.9: When hair is kelp and clothing is layered, tattered, and encrusted in sea-life. *Pirates of the Caribbean: Dead Man's Chest* (2006)

Treating all these types of simulation as just different flavors of structural dynamics is useful from both the developer's and the artist's perspective. Algorithms, data structures, and workflows developed for one type of simulation are often directly applicable to the others. Similarly, an artist's expertise in one area of

structural simulations is useful for most of them. Over the years we have successfully blurred the distinction between the flavors of structural simulations at every level, from the simulation engine, to the UI, setup, and workflows used by the production. Simulations that were once incompatible can now be used interchangeably, or can be layered upon each other with one way interactions, or can be used together.

The survey paper [WBK+07] describes many options when it comes to the choice of a dynamic mesh/rig, that is, the configuration of point-masses and springs that best represents the motion of hair and strands. Similarly, we do not have a single answer as well. A simple 1D rig was used with great success in 1998 for the hair on the *Mighty Joe Young*. Although lacking sophisticated collision and hair-hair interaction handling, and despite the fact that today we can simulate thousands of hairs like that at interactive speeds, the visual impact of those particular shots has not been significantly surpassed to date.

2D strips of cloth have also been used successfully to represent tufts of hair. While still the best solution for hair made of kelp, cloth strips cannot support their weight or resist wind equally in all directions and are, thus, not a good general solution.

We now commonly use 3D meshes of various configurations and complexities to surround and support the guide hairs during simulation. Given that we simulate only a subset of final hairs, mainly just the guide hairs, it is important to note that these meshes are more representative of a tuft of hair than of a single strand. In this context, hair-body and hair-hair interactions are closer to tuft-body and tuft-tuft interactions. As mentioned before, fine details of tufting and wisping are handled procedurally at the hair generation stage.

Finally, if absolute control and support is needed because strands become a part of and artistically driven performance, as it is the case with Davy Jones tentacles, articulated rigs with rigid constraints and motorized torque joints are used.

With regard to the choice of numerical integration technique, the common wisdom seems to point towards implicit methods. Their unconditional stability is greatly appreciated when simulations must not fail, as is the case in the games industry and real time systems, or when, for whatever reason, taking small timesteps is not an option. Still, implicit methods come at a very significant cost, as discussed in the next section. For many years we have relied exclusively on explicit integration methods, such as *Verlet* or *Newmark central differencing* [KANB03]. The visual quality of simulations that ILM has achieved using strictly explicit methods is

still unsurpassed in each category: hair, cloth, flesh, and rigid dynamics. For our deformable simulations, we now use a hybrid method that is still explicit in nature but uses full implicit integration of damping forces [BMF03]. Finite elements methods are also available to our users with a switch of a button. Their promise is in more accurate modeling of physical material properties. But they are still significantly slower than other methods and are rarely used in production.

Our lack of commitment to any particular solution is most evident in handling collisions and interactions between dynamic objects. Collisions are often the most expensive part of simulation. Our users often need the cheapest method that does the job for any particular shot or effect. We therefore offer:

- Force based point-point and point-face repulsions, including self repulsion. They are reasonably cheap (with a judicious use of bounding box trees and k-d trees) but not robust.
- Collisions of points against discretized level set volumes, with the option to handle stacking of rigid bodies [GBF03]. They are fairly robust and very fast once the volumes are computed. But computation of collision volumes can be arbitrarily slow, depending on the desired level of detail.
- Geometric collisions between moving points and moving faces, and between moving edges, with iterations to resolve multiple collisions. This is the most robust and by far the slowest option.

Each of these methods presents its own implementation challenges, particularly when it comes to achieving robustness and efficiency, but also in dealing with static friction and the details of the time differencing method. This topic could fill a course on its own and is, sadly, beyond the scope of this presentation.

Any combination of the above methods can be used in our system to handle interactions of hair meshes with other objects in the scene, including other hair meshes.

5.2.3 Challenges

The main consideration in feature production is whether the results look good. No matter how physically correct a simulation is, if it falls short of a specific artistic goal, it is not good enough. Fortunately, this does not imply that the results always have to be perfect. It means that the artists need a variety of tools and that some combination of these tools can be used to achieve the desired effect. It is actually

quite rare that the results of a simulation end up in the movie theater without being touched in some way by the artist.

An equally important consideration is that of the economy: whether the benefits of using a particular technique outweigh the costs. In an ever more competitive environment movies are done on very tight schedules and budgets. Our ability to meet the time and resource constraints depends on tools that are not only capable of achieving a variety of desired results but can do it in a predictable, reliable, and controllable fashion.

Most of our tools are designed for a long term use - they are supposed to outlive any particular production and work on the next movie, and the one after. When we are fortunate to work on effects that have never been done before, for which no tools exist, our initial attempts can be somewhat raw, requiring unusual amounts of "manual" labor from the artists, and still be useful to production. But to remain viable, the use of these new tools has to become more or less routine by the next production cycle.

Our dynamics, hair styling, and lighting tools are modes inside Zeno, ILM's large 3D application framework. There are great benefits to having such an integrated applications, from code sharing to the commonality of the UI and the workflows. Fitting our solutions into this large framework and making them work nicely with the other components also requires great care.

Here are some of the problems related to our hair simulation pipeline, but not limited to it, that we face daily:

- *Control:* Animators need tools to take the animation in any desired direction without destroying the illusion of physical correctness. When on occasion they are asked to do the physically impossible, it is a toolmaker's job as much as artist's to make the impossible look good. This is a control issue and is as important as the physical and numerical soundness of algorithms at the core of our engine.

  As described in [KANB03], control is ideally achieved with spring-like forces. "Ideally" because our system is already very well tuned to deal with such forces. However, when precise control is needed or when we know the desired simulation outcome, forces are often insufficient as they are balanced by the solver against all other events in the scene. In such cases we need to adjust the state of the system (positions, velocities, orientations, and angular momenta) directly. And in doing so we need to preserve the

stability of the system and as much of physical correctness as possible. It is really "interesting" when multiple controls want to modify the state simultaneously - this usually conflicts with our other strong desire to keep controls modular and independent from each other.

- *Robustness:* This is primarily about stability and accuracy. While a distinction between these two terms is extremely important for the developers, it is completely lost on the users. Whether the simulation fails because of a numerical overflow (stability of explicit integration methods) or it looks completely erratic or overdamped because of large errors (accuracy of implicit methods) is not nearly as important to the users as is the fact that they have nothing to show in the dailies.

  This problem is compounded by our need to provide simulation controls. Stability of our integration methods holds only as long as we do not violate the premises on which it is based. Stability analysis is difficult enough for a well formulated set of differential equations. How do we analyze stability of 10 lines of C++ code?

- *Workflow and setup issues:* Structural dynamics is just a subset of tools that artists responsible for creature development use daily. It is common that a creature requires several layers of cloth simulations, hair simulation, skin and flesh simulation, and rigid simulations for all the accessories. And there could be a dozen of them in a shot. All this in an environment where models and animation are developed concurrently and change daily.

  Zeno, our 3D application, provides a framework that makes this possible. It is by no means easy.

- *Simulation time:* No one has yet complained that simulations run too fast. Quick turnaround is extremely important to our users' productivity and to their ability to deliver a desired effect. Simulation speeds ranging from interactive to several seconds per frame are ideal. Several minutes per frame is near the limit of tolerance. Structural simulations that fail to finish overnight for the entire length of the shot are not acceptable.

- *Cost of development and support:* Algorithms that are very difficult to get right or that work only under very specific hard to meet conditions are usually less than ideal for a production environment. We do not take this argument too far because it would disqualify all of dynamics - it is a balance that we strive for. Systems based on implicit integration methods tend to be

considerably more expensive in this regard.

- Requirement for very *specialized knowledge* or experience, limiting the pool of available artists and developers.

Many of the above issues could motivate a Siggraph course on their own. It is not without regret that we touch upon them so lightly in this presentation.

Obviously, dynamics systems also provide great benefits by enabling creation of a physically believable animation that would be difficult, tedious, or impractical to do otherwise.

This cost-benefit analysis may put us on a somewhat different course from the purely research and academic community. A proof of concept, a functional prototype, a research paper is often just a starting point for us. Majority of our effort is spent on building upon proven techniques and turning them into production worthy tools - on bridging the gap between successful technology and the arts. Practical and engineering challenges of doing that easily match those of inventing the techniques in the first place. Working on these issues in a production environment, continuously balancing the needs and the abilities, has been a humbling experience for everyone involved.

5.2.4 Examples

The extreme motion of the creatures and the transformations between human and fantastic forms, e.g. werewolves tearing out from inside the skin of their human characters and vice versa, were the two biggest challenges for digital hair in the making of *Van Helsing* (2004). It was also the first time that we modeled very long curly human hair (figure 5.10) and simulated it through a full range of motion and well into the realm of humanly impossible (figures 5.11 and 5.12).

Modeling (styling) and simulation of hair was done on a smaller number of "guide" hairs – up to several hundred on a human head and almost nine thousand on the Werewolves. Before rendering, the full body of hair was created by a complex interpolation technique that also added irregularities and took care of tufting and fine wisping. These generated hairs, numbering in the mid tens of thousands for human hair and up to several million for the wolves (figure 5.13), were then passed to RenderMan for rendering as B-spline curves.

Figure 5.10: Aleera, one of the Vampire Brides from the *Van Helsing* (2004)

We relied heavily on numerical simulations to achieve a believable motion of hair. Slow-moving creatures and motion-captured humans presented very few problems. Fast moving werewolves and vampire brides were more difficult, particularly for long hair. The creatures were often animated to the camera and did not necessarily follow physically plausible 3D trajectories. In many cases the script just asked for the physically improbable. Consequently, our simulations also employed controls that were not based on physics. Particularly useful were those for matching the animation velocities in a simulation. Still, the animation sometimes had to be slowed down, re-timed, or edited to remove sharp accelerations. Wind sources with fractal variation were also invaluable for achieving realistic fast motion of hair and for matching live action footage.

Our proprietary hair pipeline was reimplemented for "Van Helsing" to allow for close integration of interactive hair placement and styling tools, hair simulation tools, and hair interpolation algorithms. The hair and (tearing) cloth dynamics systems were merged for the needs of Werewolf transformation shots in which the hair was simulated either on top of violently tearing skin, or just under it.

Figure 5.11: OpenGL rendering of a vampire bride hair simulation from the *Van Helsing* (2004)

Figure 5.12: Final composite of the simulation in figure 5.11

This integration enabled the artists to style the hair, set up and run skin and hair simulations, and sculpt post-simulation corrective shapes in a single application framework.

These same tools were simultaneously used in the making of *The Day After Tomorrow* (2004) for the CG wolves (figures 5.14, 5.15) and have since been used on many other shows for digital doubles (for example figure 5.16), a CG baby, hairy and grassy creatures, lose threads of clothing, etc.

Pirates of the Caribbean: Dead Man's Chest features Davy Jones (figures 5.8 and 5.17), an all CG character, whose most outstanding feature is his beard, designed to look like an octopus with dozens of tentacles. The beard presented multiple

Figure 5.13: A Werewolf from *Van Helsing* (2004)

challenges for animation, simulation, and deformation.

To make this character believable, it was critical that the tentacle behaved like that of an octopus, but still presented the dynamic motion of the character's performance. ILM utilized Stanford's PhysBAM simulation system as a base for the articulated rigid body simulation that drives the performance of the tentacles. Along with Stanford's solver, ILM created animation controllers that allowed the artists to direct the performance of the simulation. By incorporating the Stanford solver into our proprietary animation package, Zeno, we made it possible for a team of artists to quickly produce the animation for the 200+ shots required by production.

An articulated rigid body dynamics engine was utilized to achieve the desired look. Each tentacle was built as a chain of rigid bodies, and articulated point joints served as a connection between the rigid bodies. This simulation was performed independently of all other simulations, and the results were placed back on an animation rig that would eventually drive a separate flesh simulation. Since Davy's beard had 46 tentacles with a total of 492 rigid bodies and 446 point joints, a controller system was needed in order to make the simulation manageable for an artist. Each tentacle had a controller to define parameters to achieve the desired dynamic behavior, which was mainly influenced by the head motion and any colliding objects. Another controller, which served as a multiplier for all individual

Figure 5.14: A test run cycle with full motion blur from *The Day After Tomorrow* (2004).

controllers, helped the artist to influence the behavior of the whole beard at once. To make the tentacles curl, the connecting point joints were motorized using a sine wave that was controlled by attributes like amplitude, frequency and time. Most dynamic parameters were set along the length of the tentacle. So, in order to automate the setting of these parameters, a 2D curve represented the length of the tentacle on the x-axis and the value of the parameter on the y-axis. Periodically some tentacles required key framed animation in order to manipulate objects in the scene. When specific interactions were required from the animation pipeline, the rigid bodies were set to follow the animation and used as collision geometry for the simulated tentacles.

The control for each joint on a tentacle was accomplished using a force-based targeting system. The targeting system calculated torques between rigid objects constrained by a joint. The goal of the targeting system was to minimize the difference between the joint's target angle and its current angle. During the progression of the simulation, the target angles for the joints were modified by the animation controller. For each joint, the targeting system calculated the difference between the target orientation and current orientation. The resulting difference produced an axis of rotation that defined a torque around which the connected rigid objects rotated. The final step was to apply the calculated torque back onto the connected rigid objects.

Figure 5.15: Final composite of the CG wolves from *The Day After Tomorrow* (2004).

A real tentacle has numerous suction cups that allow the tentacle to momentarily grab onto surfaces and let go at any time. A functionality was required, termed *Stiction*, which would automatically create connecting springs between rigid bodies, to correctly mimic this momentary grab and release. The Stiction-spring interface was implemented through a set of area maps on the underside of the tentacle, defining regions on the tentacle where springs could be created. Properties of the Stiction interface defined distances at which springs could be created or destroyed, thus displaying the correct motion of the grab and release.

5.2.5 Conclusions

Simulation of hair and strands, and dynamics in general, has been very successful but is far from a solved problem. Many of the remaining challenges are not as much about possibility but more about practicality. How do we make simulations more controllable, more intuitive, more robust, more detailed but less expensive, much faster but no less accurate? Using *Pirates of the Caribbean: Dead Man's Chest* as a point of reference, the amount of detail that we can model and render today exceeds what can be simulated by an order of magnitude. And modeling and rendering are far from solved problems on their own. As computers get more powerful and our techniques get better, the demand for simulations seems to increase with the increased possibilities - or at least, that has been our experience

Figure 5.16: Jordan digital double from *The Island* (2005)

over the past decade. We see no indication of this changing any time soon. The prospect of solving some of the remaining practical problems should be wonderfully exciting to the industry practitioners and researchers alike.

5.3 Hair Simulation at Rhythm and Hues - Chronicles of Narnia

Rhythm and Hues have been well known for various works on animal characters dating back to the Babe movie. Along with each show, the whole hair pipeline have been constantly revised, enhanced, and sometimes completely rewritten, to meet the ever increasing demand of production in dealing with hair.

For the movie, *The Chronicles of Narnia, the Lion, the Witch and the Wardrobe*, the whole hair pipeline had to be revamped in many aspects. The movie had many talking animal characters, including the majestic lion, aslan. Dealing with fur of each character presented enormous challenges on every side of pipeline. Animating fur - especially longer hairs like the mane of a lion - presented a challenge that the studio had not dealt with before. A new hair dynamics solution as well as many other tools had to be developed and the tools were extensively used to simulate motion of the hair of many such mythological characters.

Figure 5.17: Davy Jones

When the crews had a chance to see and interact with wild animals (such as a real lion!), two things were pointed out.

- Most animal fur is very stiff.
- Animal fur almost always move in clumps, apparently due to hair-hair interaction

This meant that we needed to have a stiff hair dynamics system with full control over hair-hair interaction. As any experienced simulation software developer would find, this is not a particularly pleasant situation to be in to hear something is stiff in a simulation.

5.3.1 The hair simulator

From the literature, one would find a number of choices for dealing with hair-like objects. Among those are articulated rigid body method, mass-spring (lumped particle), and continuum approach, as surveyed throughout this course note. Each method has pros and cons and one could argue one methodś advantages over others.

We decided to start with the mass-spring system since we had a working code from the in-house cloth simulator. There we started by adapting the existing particle-based simulator to hair.

mass-spring structure for hair

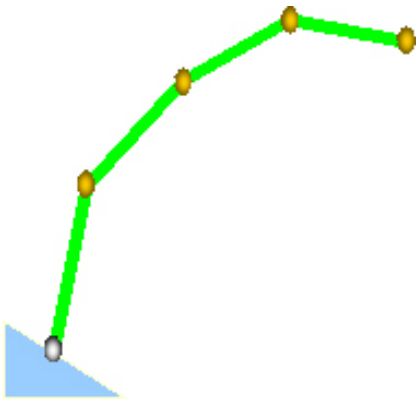

Figure 5.18: Mass spring structure for hair simulation

In our simulator, each hair would be represented by a number of nodes, each node representing the (lumped) mass of certain portion of hair. In practice, each CV of guide hairs (created at the grooming stage) was used as the mass node. Such nodes are connected by two types of springs - linear and angular springs. Linear

springs maintain the length of each hair segment and angular springs maintain the relative orientation between hair segments.

Linear spring was simply taken from the standard model used for cloth simulator, but a new spring force had to be developed for the angular springs. We considered the use of so-called flexion springs that are widely used in cloth simulation. With this scheme, each spring connects nodes that are two (or more) nodes apart.

However, after initial tests, it was apparent that this spring would not serve our purpose since there are a lot of ambiguities in this model and angles are not always preserved. This ambiguity would result in some unwanted wrinkles in the results (in the images below, all three configurations are considered the same from linear springś point of view).

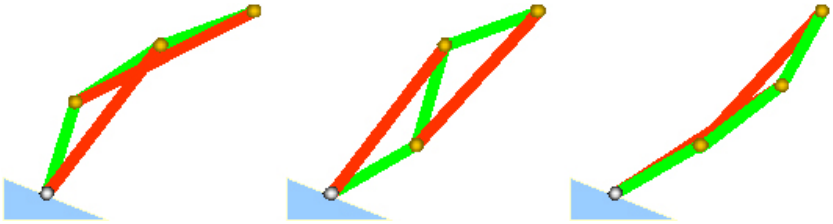

Figure 5.19: ambiguity of linear springs

Eventually, the hair angle preservation had to be modeled directly from angles. We derived the angle preservation force by first defining an energy term defined on two relative angles between hair segments, and then by taking variational derivatives to derive forces. A matching damping force was added as well.

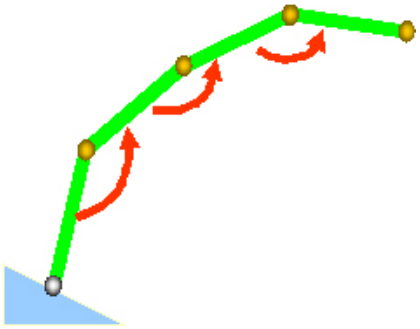

Figure 5.20: angular springs

Derivation on angles are usually far more difficult than working on positions, and it would also require additional data such as anchor points attached to the root such that angles could be computed at the root point as well. To compute a full angle around each node, each node would have an attached axis that was generated at the root and propagated to each node . We simulated the motion of each hair along with external forces such as gravity, wind forces. The time integration was performed with a full implicit integration scheme. As a consequence, the simulator was very stable dealing with the stiff hair problem. Extremely stiff hairs (such as wire) needed some numerical treatment such as modification of jacobian matrices, etc., but in general, this new approach was made very stable and could handle very stiff hairs (almost like a wire) in a fixed time stepping scheme.

In the absence of collision and hair- hair interaction, each hair could be solved independently, and solved very fast if a direct numerical method was employed (thousands of guide hairs could be simulated in less than a second per frame). In practice, the simulation time was dominated by collision detection and hair hair interaction. Overall integrator time was only a small fraction (less than 3%).

Figure 5.21: This CG lion test was performed before the production started, as verification on many aspects such as simulation of hair-hair interaction, collision handling, grooming of hair, rendering, and compositing.

Collision Handling

There are two types of collision in hair simulation - Hair would collide against the character body, but would also collide against other hairs. These two cases were separately handled, and each case presented challenges and issues.

Collision handling between hair and characters.

For collision handling, each character was assigned as a collision object and collision of each hair against the collision object was performed using the standard collision detection techniques (such as AABB, Hashing, OBB, etc.) with some speed optimizations (e.g. limiting distance query to the length of hair, etc.) added.

If a CV was found to be penetrating the collision object, it was pushed out by a projection scheme that was tied to our implicit integrator. For most cases, the simple scheme worked very well, even in some situations where collision objects are pinched between themselves.

However, in character animation, some amount of pinching is unavoidable (especially when characters are rolling or being dragged on the ground), and the simulator had to be constantly augmented and modified to handle such special case handling of user error in collision handling.

For example, in some scenes, hair roots often lie deep under the ground. In such cases, applying standard collision handler would push things out to the surface, but hair had to be pulled back since the root had to lie under the ground.

This would eventually result in oscillations and jumps in the simulation. Our simulator had additional routines to detect such cases and provided options to freeze the simulation for the spot or to blend in simulation results. In addition, animations were adjusted (mostly for aesthetical reasons) and other deformation tools were also employed to sculpt out the problem area in collision.

Hair-hair interaction

Early on, it was determined that the ability to simulate hair-hair interaction was a key feature that we wanted to have. Without hair-hair interaction, hairs would

simply penetrate through each other, and one would lose the sense of volume in hair simulation.

This was especially important since our hair simulation was run on guide hairs, and each guide hair could represent a certain amount of volumes around it. So, the sense of two volumes interacting with each other was as important as simulating each guide hair accurately.

Having based our simulator on a mass-spring model, we added the hair interaction effect as additional spring force acting on hair segments. Whenever a hair is close to another hair, a spring force was temporarily added to prevent nearby hairs from further approaching each other, and also to repel too close ones away from each other. The amount of spring force was scaled by such factors as distance, relative velocity, and user-specified strength of hair interaction.

Adding wind effect

In the movie, many scenes were shot in extremely windy environment. There was almost always some amount of wind in the scene, whether it was a mild breeze or gusty wind. Once we had a working simulator, the next challenge was to add these wind effects with full control.

In general, hair simulation was first run on (only) thousands of guide hairs and then the guide hairs would drive motion of millions of hairs that were finally rendered. Correspondingly, there were two controls over the wind effects.

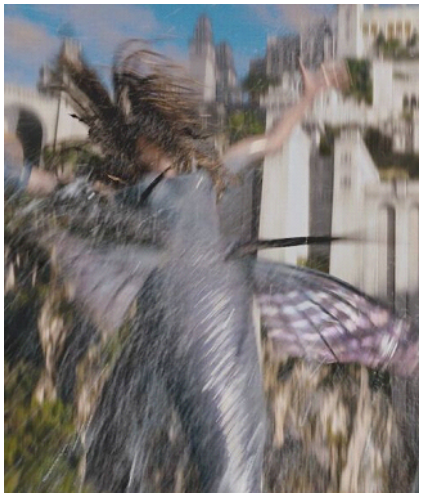

Figure 5.22: Simulating hair hair interaction for mermaid.

First, dynamics had a wind force that applies random and directional noise-driven force that would move guide hairs.

Second, a tool called *pelt wind* was developed and added on top of the dynamics motion, providing subtle control over motion in every rendered hair

Bad inputs / user errors

Throughout the production, we would battle issues with bad inputs to the simulator. In practice, inputs to simulation are never perfect sometimes there would be two hairs stemming from exactly the same root position, sometimes hair shape was too crooked. In some cases, hairs were too tangled to begin with, and hair interaction alone could not handle the situation.

Additional hair model processing tools were then used to tame the input such as untangling hair orientation more evenly or straightening out crooked hairs. In the end, the simulator was also used as a draping tool that users could use to process and clean up some of the hand modeled hair geometry.

5.3.2 Layering and mixing simulations

Often, digital counterpart of real actors were used and mixed into the real scenes. Simulations were also used for clothing of such characters (such as cape, skirt, etc.) and even skins of winged characters.

At times, cloth simulation and hair simulation had to work together. Cloth would collide against hairs, but hair would in turn collide with cloth. In such cases, a proxy geometry was built to represent the outer surface of hair volume. Cloth would then collide against the proxy geometry and then served as collision object for hair simulation.

This concept of simulation layering was used all over. For some hair simulation, cloth was first simulated as the proxy geometry for hair, and then hair simulation was run, roughly following the overall motion driven by the proxy geometry, and then individual hair motion and hair interaction was added.

5.3.3 Simulating flexible objects for crowd characters

In addition to hero characters that had 2-3 hair / cloth simulations per character, the whole army of characters had to be animated, and consequently their cloth, hair, and anything soft had to be simulated. As an example, the scene in Figure 5.23 shows 20+ hero characters, and all the background (crowd) characters were

given another pass of simulation, to give their banner, armor, flag, and hair flowing looks.

Figure 5.23: Battle scene.

The simulator used for crowd characters was adapted from our in-house cloth simulator, and modified to meet the new requirements. For distance characters, geometry used for crowd simulation was of relatively low resolution (¡200 polys). The simulator had to not only run fast, but also had to give convincing results on such low resolution geometry.

Many characters in crowd shots are not visible until certain moments in frames, and also change its visual importance as they move in and out of the camera. This fact was extensively exploited in our simulation level of detail system.

In contrast to conventional simulation system where a simulator computes an end-to-end frame calculation, we simulated all the characters at each frame, and constantly examined whether some characters were moving out of the camera frustrum. For such invisible characters, the lowest level of detail of used in simulation

. On the other hand, as characters move closer to the camera, the detail was promoted and more time was spent on simulating higher resolution version of the geometry. This way, we could keep the desired fidelity in motion, while minimizing the requirements for computational resources.

The framework required that simulation had to be interchangeable between different resolutions, so special attention and care had to be paid to ensure that the

solverś state carries over from lower resolution to higher resolution without noticeable jump or discontinuity in motion.

Typically, several cloth simulations were run per each character, some cloth patches representing a strip of hair (we did not run hair simulator directly on any crowd character) that actual hair geometry would be attached at the render time. About 3 to 4 different resolutions were used and switched during simulation. For example, a characterś hair would be simulated as a simple polygon strip at the lowest level, and then refined all the way up to 20-100 strips representing the same geometry in much higher detail.

Bibliography

[ALS98] C. Adler, J. Lock, and B. Stone. Rainbow scattering by a cylinder with a nearly elliptical cross section. *Applied Optics*, 37(9):1540–1550, 1998.

[AP07] B. Audoly and Y. Pomeau. *Elasticity and Geometry: from hair curls to the nonlinear response of shells*. Oxford University Press, À paraître en 2007.

[AUK92a] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*. ACM SIGGRAPH, 1992.

[AUK92b] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'92 conference)*, pages 111–120, August 1992.

[BAC+06] 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 ACM SIGGRAPH'06 conference)*, pages 1180–1187, August 2006.

[Ban94] David C. Banks. Illumination in diverse codimensions. *Proc. of ACM SIGGRAPH*, 1994.

[BAQ+05] 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, *Eurograph-*

ics'05 (short papers). Eurographics, August 2005. Eurographics'05 (short papers).

[Bar92] David Baraff. *Dynamic simulation of non-penetrating rigid bodies*. PhD thesis, Department of Computer Science, Cornell University, March 1992.

[Bar96] David Baraff. Linear-time dynamics using lagrange multipliers. *Proceedings of SIGGRAPH 96*, pages 137–146, August 1996.

[BCN03] Y. Bando, B-Y. Chen, and T. Nishita. Animating hair with loosely connected particles. *Computer Graphics Forum*, 22(3):411–418, 2003. Proceedings of Eurographics'03.

[BCP96] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Classics in Applied Mathematics. SIAM, 1996.

[Ber06] F. Bertails. *Simulation de chevelures naturelles*. PhD thesis, Institut National Polytechnique de Grenoble, 2006.

[BKCN03a] 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*, pages 207–213, July 2003.

[BKCN03b] 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 *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, aug 2003.

[BMC05] F. Bertails, C. Ménier, and M-P. Cani. A practical self-shadowing algorithm for interactive hair animation. In *Proc. Graphics Interface*, pages 71–78, May 2005.

[BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Eurographics/SIGGRAPH Symposium on Computer Animation*, San Diego, California, 2003.

[Bru99] A. Bruderlin. A method to generate wet and broken-up animal fur. In *Computer Graphics and Applications, 1999. Proceedings. Seventh Pacific Conference*, pages 242–249, October 1999.

[BS91] H. Bustard and R. Smith. Investigation into the scattering of light by human hair. *Applied Optics*, 24(30):3485–3491, 1991.

[BW92] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'92 conference)*, pages 303–308, 1992.

[BW98a] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proc. of ACM SIGGRAPH*, pages 43–54, 1998.

[BW98b] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54, July 1998.

[CCK05a] B. Choe, M. Choi, and H-S. Ko. Simulating complex hair with robust collision handling. In *ACM SIGGRAPH - EG Symposium on Computer Animation*, pages 153–160, August 2005.

[CCK05b] Byoungwon Choe, Min Gyu Choi, and Hyeong-Seok Ko. Simulating complex hair with robust collision handling. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 153–160, 2005.

[CJY02a] J. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. In *ACM SIGGRAPH - EG Symposium on Computer Animation*, pages 73–80, July 2002.

[CJY02b] Johnny Chang, Jingyi Jin, and Yizhou Yu. A practical model for mutual hair inteactions. In *Proceedings of Symposium on Computer Animation*. ACM SIGGRAPH, San Antonio, USA, July 2002.

[CK05] B. Choe and H-S. Ko. A statiscal wisp model and pseudophysical approaches for interactive hairstyle generation. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):153–160, March 2005.

[CPS92] R.W. Cottle, J. S. Pang, and R.E. Stone. *The linear complementarity problem*. Academic Press, 1992.

[CSDI99] L. Chen, S. Saeyor, H. Dohi, and M. Ishizuka. A system of 3d hairstyle synthesis based on the wisp model. *The Visual Computer*, 15(4):159–170, 1999.

[DKY+00] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita. A simple, efficient method for realistic animation of clouds. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'00 conference)*, pages 19–28. ACM Press/Addison-Wesley Publishing Co., 2000.

[DMTKT93] A. Daldegan, N. Magnenat-Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum*, 12(3):211–221, 1993.

[Dur04] D. Durville. Modelling of contact-friction interactions in entangled fibrous materials. In *Procs of the Sixth World Congress on Computational Mechanics (WCCM VI)*, September 2004.

[Fea87] Roy Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[GBF03] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics (SIGGRAPH Proceedings)*, 2003.

[Gol97] D. Goldman. Fake fur rendering. In *Proceedings of ACM SIGGRAPH'97*, Computer Graphics Proceedings, Annual Conference Series, pages 127–134, 1997.

[GZ02] Y. Guang and H. Zhiyong. A method of human short hair modeling and real time animation. In *Proceedings of Pacific Graphics'02*, pages 435–438, September 2002.

[Had03] Sunil Hadap. *Hair Simulation*. PhD thesis, MIRALab, CUI, University of Geneva, January 2003.

[HKS98] T. Hou, I. Klapper, and H. Si. Removing the stiffness of curvature in computing 3-d filaments. *J. Comput. Phys.*, 143(2):628–664, 1998.

[HMT01a] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338, 2001. Proceedings of Eurographics'01.

[HMT01b] Sunil Hadap and Nadia Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338, 2001.

[HS98] W. Heidrich and H.-P. Seidel. Efficient rendering of anisotropic surfaces using computer graphics hardware. *Proc. of Image and Multi-dimensional Digital Signal Processing Workshop (IMDSP)*, pages 315–318, 1998.

[JLD99] H. Jensen, J. Legakis, and J. Dorsey. Rendering of wet material. *Rendering Techniques*, pages 273–282, 1999.

[KANB03] Z. Kačić-Alesić, M. Nordenstam, and D. Bullock. A practical dynamics system. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 7–16. Eurographics Association, 2003.

[KAT93] T. Kurihara, K. Anjyo, and D. Thalmann. Hair animation with collision detection. In *Proceedings of Computer Animation'93*, pages 128–138. Springer, 1993.

[Ken04] Erleben Kenny. *Stable, Robust, and Versatile Multibody Dynamics Animation*. PhD thesis, Department of Computer Science, University of Copenhagen, November 2004.

[KH00] C. Koh and Z. Huang. Real-time animation of human hair modeled in strips. In *EG workshop on Computer Animation and Simulation (EG CAS'00)*, pages 101–112, September 2000.

[KH01] C. Koh and Z. Huang. A simple physics model to animate human hair modeled in 2D strips in real time. In *EG workshop on Computer Animation and Simulation (EG CAS'01)*, pages 127–138, September 2001.

[KHS04] M. Koster, J. Haber, and H-P. Seidel. Real-time rendering of human hair using programmable graphics hardware. In *Computer Graphics International (CGI'04)*, pages 248–256, June 2004.

[Kim02] Tae-Yong Kim. *Modeling, Rendering, and Animating Human Hair*. PhD thesis, University of Southern California, 2002.

[KK89] J. Kajiya and T. Kay. Rendering fur with three dimensional textures. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'89 conference)*, Computer Graphics Proceedings, Annual Conference Series, pages 271–280, New York, NY, USA, 1989. ACM Press.

[KN99] W. Kong and M. Nakajima. Visible volume buffer for efficient hair expression and shadow generation. In *Computer Animation*, pages 58–65. IEEE, 1999.

[KN00] T.-Y. Kim and U. Neumann. A thin shell volume for modeling human hair. In *Computer Animation 2000*, IEEE Computer Society, pages 121–128, 2000.

[KN01] T-Y. Kim and U. Neumann. Opacity shadow maps. In *Rendering Techniques 2001*, Springer, pages 177–182, July 2001.

[KN02] T-Y. Kim and U. Neumann. Interactive multiresolution hair modeling and editing. *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH'02 conference)*, 21(3):620–629, July 2002.

[Kok04] Evangelos Kokkevis. Practical physics for articulated characters. In *Proceedings of Game Developers Conference 2004*, 2004.

[LFHK88] B. Lindelof, B. Forslind, MA. Hedblad, and U. Kaveus. Human hair form. morphology revealed by light and scanning electron microscopy and computer aided three-dimensional reconstruction. *Arch. Dermatol.*, 124(9):1359–1363, 1988.

[LGLM00] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Distance queries with rectangular swept sphere volumes. *Proc. of IEEE Int. Conference on Robotics and Automation*, pages 24–48, 2000.

[LK01a] D-W. Lee and H-S. Ko. Natural hairstyle modeling and animation. *Graphical Models*, 63(2):67–85, March 2001.

[LK01b] Doo-Won Lee and Hyeong-Seok Ko. Natural hairstyle modeling and animation. *Graphical Models*, 63(2):67–85, March 2001.

[LTT91] A. M. LeBlanc, R. Turner, and D. Thalmann. Rendering hair using pixel blending and shadow buffers. *The Journal of Visualization and Computer Animation*, 2(3):92–97, – 1991.

[LV00] T. Lokovic and E. Veach. Deep shadow maps. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'00 conference)*, pages 385–392. ACM Press/Addison-Wesley Publishing Co., 2000.

[Mar74] D. Marcuse. Light scattering from elliptical fibers. *Applied Optics*, 13:1903–1905, 1974.

[Mir96] Brian Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley, December 1996.

[Mit96] D. Mitchell. Consequences of stratified sampling in graphics. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'96 conference)*, pages 277–280, 1996.

[MJC+03a] S. Marschner, H. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH'03 conference)*, 22(3):281–290, July 2003.

[MJC+03b] S. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. *ACM Transactions on Graphics*, 22(3):780–791, July 2003. Proceedings of ACM SIGGRAPH 2003.

[MKBR04] T. Mertens, J. Kautz, P. Bekaert, and F. Van Reeth. A self-shadow algorithm for dynamic hair using density clustering. In *Proceedings of Eurographics Symposium on Rendering*, pages 173–178, 2004.

[MTM98] C. Mount, D. Thiessen, and P. Marston. Scattering observations for tilted transparent fibers. *Applied Optics*, 37(9):1534–1539, 1998.

[NdP98] I. Neulander and M. Van de Panne. Rendering generalized cylinders with paintstrokes. In *Graphics Interface*, pages 233–242, 1998.

[Ney98] F. Neyret. Modeling animating and rendering complex scenes using volumetric textures. *IEEE Transaction on Visualization and Computer Graphics*, 4(1):55–70, Jan-Mar 1998.

[NR01] O. Nocent and Y. Remion. Continuous deformation energy for dynamic material splines subject to finite displacements. In *EG workshop on Computer Animation and Simulation (EG CAS'01)*, pages 88–97, September 2001.

[OBKA03] H. Ono, S. Benza, and Z. Kačić-Alesić. Bringing digital crash dummies to life for 'the hulk'. In *SIGGRAPH Sketches and Applications*, San Diego, California, 2003. ACM SIGGRAPH.

[Ove91] C. Van Overveld. An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation. *The Visual Computer*, 7:29–38, 1991.

[Pai02a] D. Pai. Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum*, 21(3):347–352, 2002. Proceedings of Eurographics'02.

[Pai02b] D. K. Pai. Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum*, 21(3):347–352, 2002.

[PCP01a] E. Plante, M-P. Cani, and P. Poulin. A layered wisp model for simulating interactions inside long hair. In *EG workshop on Computer Animation and Simulation (EG CAS'01)*, Computer Science, pages 139–148. Springer, September 2001.

[PCP01b] Eric Plante, Marie-Paule Cani, and Pierre Poulin. A layered wisp model for simulating interactions inside long hair. In *Proceedings of Eurographics Workshop, Computer Animation and Simulation*. EUROGRAPHICS, Manchester, UK, September 2001.

[PCP02] E. Plante, M-P. Cani, and P. Poulin. Capturing the complexity of hair motion. *Graphical Models (Academic press)*, 64(1):40–58, january 2002. submitted Nov. 2001, accepted, June 2002.

[PK96] F. C. Park and I. G. Kang. Cubic interpolation on the rotation group using cayley parameters. In *Proceedings of the ASME 24th Biennial Mechanisms Conference*, Irvine, CA, 1996.

[PS03] Jong-Shi Pang and David E. Stewart. Differential variational inequalities. Technical report, http://www.cis.upenn.edu/davinci/publications, 2003.

[PT96] J. S. Pang and J. C. Trinkle. Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction. *Mathematical Programming*, 73:199–226, 1996.

[QT96] H. Qin and D. Terzopoulos. D-nurbs: A physics-based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, 1996.

[RCFC03] L. Raghupathi, V. Cantin, F. Faure, and M.-P. Cani. Real-time simulation of self-collisions for virtual intestinal surgery. In Nicholas Ayache and Hervé Delingette, editors, *Proceedings of the International Symposium on Surgery Simulation and Soft Tissue Modeling*, number 2673 in Lecture Notes in Computer Science, pages 15–26. Springer-Verlag, 2003.

[RCT91a] R. Rosenblum, W. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *The Journal of Visualization and Computer Animation*, 2(4):141–148, October-December 1991.

[RCT91b] R. Rosenblum, W. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: Modeling, rendering, and animation. *The Journal of Visualization and Computer Animation*, 2(4):141–148, 1991.

[RGL05a] S. Redon, N. Galoppo, and M. Lin. Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics (Proceedings of the ACM SIGGRAPH'05 conference)*, 24(3):936–945, 2005.

[RGL05b] Stephane Redon, Nico Galoppo, and Ming C. Lin. Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics*, 24(3):936–945, aug 2005.

[RJFdJ04] José Ignacio Rodríguez, José Manuel Jiménez, Francisco Javier Funes, and Javier García de Jalón. Recursive and residual algorithms for the efficient numerical integration of multi-body systems. *Multibody System Dynamics*, 11(4):295–320, May 2004.

[Rob94] Clarence R. Robbins. *Chemical and Physical Behavior of Human Hair*. Springer-Verlag, New York, third edition, 1994.

[Rob02] C. Robbins. *Chemical and Physical Behavior of Human Hair*. 4th ed. Springer, 2002.

[Rub00] M.B. Rubin. *Cosserat Theories: Shells, Rods and Points*. Springer, 2000.

[SGF77] Robert F. Stamm, Mario L. Garcia, and Judith J. Fuchs. The optical properties of human hair i. fundamental considerations and goniophotometer curves. *J. Soc. Cosmet. Chem.*, 28:571–600, 1977.

[Sha01] Ahmed A. Shabana. *Computational Dynamics*. Wiley-Interscience, 2001.

[Tri70] R. Tricker. *Introduction to Meteorological Optics*. Mills & Boon, London, 1970.

[WBK+07] K. Ward, F. Bertails, T.-Y. Kim, S. Marschner, M.-P. Cani, and M. Lin. A survey on hair modeling: styling, simulation, and rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):213–234, Mar/Apr. 2007.

[WGL04] K. Ward, N. Galoppo, and M. Lin. Modeling hair influenced by water and styling products. In *Proceedings of Computer Animation and Social Agents (CASA'04)*, pages 207–214, May 2004.

[WGL06] K. Ward, N. Galoppo, and M. Lin. A simulation-based vr system for interactive hairstyling. In *IEEE Virtual Reality - Application and Research Sketches*, pages 257–260, 2006.

[WGL07] K. Ward, N. Galoppo, and M. Lin. Interactive virtual hair salon. In *PRESENCE to appear (June 2007)*, 2007.

[WL03] K. Ward and M. Lin. Adaptive grouping and subdivision for simulating hair dynamics. In *Proceedings of Pacific Graphics'03*, pages 234–243, September 2003.

[WLL+03] K. Ward, M. Lin, J. Lee, S. Fisher, and D. Macri. Modeling hair using level-of-detail representations. In *Proceedings of Computer Animation and Social Agents (CASA'03)*, pages 41–47, May 2003.

[Wol99] S. Wolfram. *The Mathematica book (4th edition)*. Cambridge University Press, New York, NY, USA, 1999.

[WS92] Y. Watanabe and Y. Suenaga. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics and Applications*, 12(1):47–53, 1992.

[WW90] A. Witkin and W. Welch. Fast animation and control of non-rigid structures. In *Computer Graphics Proceedings (Proceedings of the ACM SIGGRAPH'90 conference)*, pages 243–252, 1990.

[YXYW00] X. Yang, Z. Xu, J. Yang, and T. Wang. The cluster hair model. *Graphics Models and Image Processing*, 62(2):85–103, March 2000.

[ZFWH04] C. Zeller, R. Fernando, M. Wloka, and M. Harris. Programming graphics hardware. In *Eurographics - Tutorials*, September 2004.

[Zvi86] C. Zviak. *The Science of Hair Care*. Marcel Dekker, 1986.