Modeling Hair using Levels of Detail

by
Kelly Anne Ward

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2005

Approved by:
Ming C. Lin, Advisor
Dinesh Manocha, Reader
Mark Foskey, Reader
Angelmo Lastra, Committee Member
Gary Bishop, Committee Member
ABSTRACT

KELLY ANNE WARD: Modeling Hair using Levels of Detail.
(Under the direction of Ming C. Lin.)

Modeling hair is important for creating realistic virtual humans in various applications. A human head typically contains over 100,000 strands of hair, each strand being extremely thin and thereby generating an intricate hair volume. Due to its complex nature, hair simulation, including the reproduction of interactions both among the hair strands and between the hair and the avatar, is computationally overwhelming. The rendering of hair is similarly challenging, particularly as a result of the shadows caused by hair self-occlusions. Consequently, many interactive applications in practice today are forced to overlook several complex features of hair in order to attain a desired performance. By simplifying the hair volume, these applications often compromise the visual quality of the hair. Moreover, they typically contain a considerable amount of unnecessary computation allocated towards strands of hair that have minimal significance to such applications.

In this thesis, I introduce the use of levels of detail for modeling hair. Levels of detail enable a simulation to allocate the majority of computational resources towards modeling those strands of hair with the most significance to the application at hand. The methods I discuss are based on the use of three discrete hair representations: strips, clusters and strands. Each representation provides a different level of visual fidelity and performance speed for simulating and rendering hair. The visibility, motion and viewing distance of the hair, as well as the user’s interaction, are considered in identifying those groups of hair with the greatest significance to the application. The techniques I present then accelerate the simulation and rendering of the hair strands with the lowest importance to the application, thereby accelerating the overall modeling of the hair.

Moreover, in this dissertation I offer several techniques for dynamically changing the physical structure, behavior and appearance of hair as water or styling products are applied to it. These styling methods are then coupled with the level of detail framework to allow
users to interactively style virtual hair.
ACKNOWLEDGMENTS

There are many people whose help and encouragement have been instrumental in my completion of this dissertation. I have been very fortunate to have had so much support during this time and I would like to extend my heartfelt thanks to each of these people.

First, I owe a great thanks to my adviser, Prof. Ming Lin. Not only has she given me the opportunity, advice and direction for pursuing my research but her tireless dedication to her students has been invaluable to me finishing this dissertation. I would also like to thank Prof. Dinesh Manocha. I began this research through course projects in classes led by Prof. Manocha and he has always lent valuable input and encouragement towards my work. Moreover I would like to thank the rest of my committee members, Prof. Mark Foskey, Prof. Anselmo Lastra and Prof. Gary Bishop, for their feedback and dedication of time concerning this work.

This research has been a rewarding yet laborious endeavor. I am very grateful to the individuals who have collaborated with me on this work, making the entire process as painless as possible. In particular I would like to thank Nico Galoppo who had a large contribution towards the graphics hardware implementation of hair rendering and who also created the hairdryer and spray bottle models used in the virtual hair salon. Thank you also to Joohi Lee, Susan Fisher and Dean Macri for their contributions towards this work over the years.

The avatar models and human motion generation used in many demonstrations were created using Poser software, which was introduced to me by Andrei State and Ben Lok. The avatars and motion were significant contributions to the quality of the resulting videos that demonstrated the capabilities of this hair modeling framework. Moreover, I would like to thank David Marshburn for helping me incorporate the PHANToM and GHOST toolkit into the virtual hair salon. Great thanks as well to my real life hair models: Wendy Ward,
Caroline Green and Kavita Coombe. They graciously allowed me to photograph their hair for comparison purposes and for observing and measuring real results, which was enormously beneficial to my work.

I extend many thanks to all the members of the GAMMA research group, past and present. Their valuable feedback and research discussions over the past several years have been tremendously helpful. Similarly, I am grateful to the entire faculty, students and staff of the UNC Computer Science Department for creating a friendly and cooperative working environment to which I am thankful to have been a part.

I would like to acknowledge the individuals who collaborated with me on a survey paper on hair research, Florence Bertails, Tae-Yong Kim, Steve Marschner and Marie-Paule Cani, which constituted a large portion of the text that appears in Chapter 2.

Furthermore I would like to acknowledge the funding agencies that have supported this research over the past few years, Intel Corporation, Army Research Office, National Science Foundation and Office of Naval Research.

My time at the University of North Carolina at Chapel Hill has been a precious experience for me. I am very grateful to the wonderful friends I have made in my time here. In particular I am indebted to Caroline Green and Kavita Coombe for their support and friendship, which have meant the world to me.

Finally, I extend my deepest thanks and appreciation to my parents and my sister Wendy who have always given me nothing but love and encouragement throughout my life. They always had more confidence in me than I had in myself. I thank you from the bottom of my heart!
# Contents

List of Tables \hspace{1cm} \hspace{1cm} xv

List of Figures \hspace{1cm} \hspace{1cm} xvii

1 Introduction \hspace{1cm} 1
  1.1 Motivation \hspace{1cm} 2
  1.1.1 Problem Overview \hspace{1cm} 3
  1.1.2 Applications and Benefits \hspace{1cm} 8
  1.2 Thesis \hspace{1cm} 9
  1.3 Main Results \hspace{1cm} 10
    1.3.1 Level-of-Detail Techniques for Hair Modeling \hspace{1cm} 10
    1.3.2 Accelerated Hair Modeling Algorithms \hspace{1cm} 13
    1.3.3 Dynamically Changing Hair Properties \hspace{1cm} 15
    1.3.4 User Interaction with Dynamic Hair \hspace{1cm} 17
  1.4 Thesis Organization \hspace{1cm} 18

2 Related Work \hspace{1cm} 20
  2.1 Hairstyling \hspace{1cm} 21
    2.1.1 Hair Placement on Scalp \hspace{1cm} 22
    2.1.2 Geometry-Based Hairstyling \hspace{1cm} 24
    2.1.3 Physics-based Hairstyling \hspace{1cm} 28
    2.1.4 Generation of Hairstyles from Images \hspace{1cm} 31
  2.2 Hair Animation \hspace{1cm} 33
    2.2.1 Animation of Individual Hair Strands \hspace{1cm} 33
2.2.2 Hair as a Continuous Medium ........................................ 38
2.2.3 Hair as Disjoint Groups ............................................... 42
2.3 Hair Rendering .......................................................... 45
   2.3.1 Light Scattering in Hair ........................................... 46
   2.3.2 Representing Hair for Rendering ................................. 53
   2.3.3 Hair Self-Shadowing .............................................. 54
   2.3.4 Rendering Acceleration Techniques ............................. 57
2.4 Model Simplification .................................................... 58

3 Level-of-Detail Framework for Modeling Hair ....................... 60
   3.1 The Base Skeleton ................................................... 63
   3.2 Geometric Representations ......................................... 66
      3.2.1 Strips .......................................................... 66
      3.2.2 Clusters ....................................................... 67
      3.2.3 Strands ........................................................ 68
   3.3 Subdivision Representations ........................................ 70
   3.4 Hair Simulation ...................................................... 70
      3.4.1 Single-Skeleton Dynamics ..................................... 71
      3.4.2 Collision Detection and Response ............................ 76
   3.5 Hair Rendering ...................................................... 83
      3.5.1 Lighting Scattering ............................................ 83
      3.5.2 Hair Self-Shadowing .......................................... 84
      3.5.3 Level-of-Detail Hair Rendering ............................... 87
   3.6 Runtime Selection of Hair ......................................... 89
      3.6.1 Visibility ....................................................... 90
      3.6.2 Viewing Distance ............................................. 91
      3.6.3 Hair Motion .................................................... 92
      3.6.4 Combining Criteria ........................................... 93
   3.7 Results and Comparisons .......................................... 93
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7.1</td>
<td>Performance Comparisons</td>
<td>94</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Analysis and Discussion</td>
<td>96</td>
</tr>
<tr>
<td>3.8</td>
<td>Summary</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td>Hair Hierarchy</td>
<td>102</td>
</tr>
<tr>
<td>4.1</td>
<td>Construction of Hair Hierarchy</td>
<td>105</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Strip and Cluster Subdivision</td>
<td>106</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Strand Group Subdivision</td>
<td>108</td>
</tr>
<tr>
<td>4.2</td>
<td>Level-of-Detail Transitions</td>
<td>109</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Adaptive Subdivision</td>
<td>110</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Adaptive Merging</td>
<td>111</td>
</tr>
<tr>
<td>4.3</td>
<td>Results and Comparisons</td>
<td>112</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Analysis and Discussion</td>
<td>112</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Performance Comparisons</td>
<td>114</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Limitations</td>
<td>115</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Comparisons Against Other Approaches</td>
<td>118</td>
</tr>
<tr>
<td>4.4</td>
<td>Summary</td>
<td>119</td>
</tr>
<tr>
<td>5</td>
<td>Modeling Hair with Water and Styling Products</td>
<td>120</td>
</tr>
<tr>
<td>5.1</td>
<td>Background</td>
<td>122</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Hair and Water</td>
<td>122</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Hair and Styling Products</td>
<td>123</td>
</tr>
<tr>
<td>5.2</td>
<td>Dual-Skeleton System</td>
<td>125</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Dual-Skeleton Setup</td>
<td>126</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Dual-Skeleton Dynamics</td>
<td>128</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Localized Collision Detection</td>
<td>129</td>
</tr>
<tr>
<td>5.3</td>
<td>Modifications of Physical Properties due to Water and Styling Products</td>
<td>131</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Adjustment of Dynamics Properties</td>
<td>131</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Flexible Geometric Structure</td>
<td>133</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Dynamic Bonds between Strands</td>
<td>136</td>
</tr>
</tbody>
</table>
5.4 Rendering Wet Hair ........................................ 137
  5.4.1 Interaction of Light with Wet Hair ................. 137
  5.4.2 Capturing the Rendering Influences of Water ....... 139
  5.4.3 Results of Wet Hair ................................ 140
  5.4.4 Influence of Styling Products ...................... 140
5.5 Results and Discussion .................................. 141
  5.5.1 Analysis ........................................... 141
  5.5.2 Limitations ....................................... 143
5.6 Summary .................................................. 144

6 Interactive Hairstyling ................................... 146
  6.1 User Interface ......................................... 149
  6.2 Interactive Dynamic Simulation ....................... 150
    6.2.1 Simulation Localization ......................... 151
    6.2.2 Multi-Resolution Simulation with the Grid ....... 153
  6.3 User Interaction and Applications .................... 154
    6.3.1 Hair Cutting ..................................... 154
    6.3.2 Applying Water .................................. 156
    6.3.3 Applying Hairpsray and Mousse ................. 157
    6.3.4 Grabbing and Moving Hair ...................... 157
    6.3.5 Hairdryer ....................................... 158
  6.4 Hierarchy Inheritance .................................. 159
  6.5 Results and Discussion ................................ 160
    6.5.1 Discussion ...................................... 160
    6.5.2 Limitations and Future Work .................... 161
  6.6 Summary ............................................... 164

7 Conclusion ................................................ 167
  7.1 Summary of Results ................................... 167
  7.2 Future Work ........................................... 171
7.2.1 Partial Visibility and Graphics Hardware ........................................ 171
7.2.2 Hairstyle Creation ................................................................. 172
7.2.3 Additional Limitations and Future Work ...................................... 173
7.3 Conclusion ................................................................................. 173

Bibliography ...................................................................................... 175
List of Tables

3.1 Performance Comparison. *Simulation for a camera zooming out. The average performance numbers are measured in seconds per frame.* ........................................... 94

4.1 Performance Comparison. *Simulation for a stationary camera. The average performance numbers are measured in seconds per frame.* ....................... 114

4.2 Performance Comparison. *Simulation for a camera zooming out. The average performance numbers are measured in seconds per frame.* ....................... 115

4.3 Memory Comparison. *Approximate memory usage comparisons measured in KB for the coarsest strips, coarsest clusters, coarsest strand groups (typical wisp implementations), discrete LODs (Chapter3) and hair hierarchy implementation.* 117
List of Figures

1.1 Real hair is complex and can vary widely in style and color. ........................................... 2

1.2 Different representations for modeling hair. (left) Hair strips [KH01] (right) Hair strands or wisps [KN02]. ......................................................... 4

1.3 (top) Image from movie “Final Fantasy: The Spirits Within” 2001 (bottom) Screenshot from videogame “The Legend of Zelda: Twilight Princess” 2005. . . . 6

1.4 Simulation factors can decide how much detail is visible. From left to right (1) Camera is close to hair making it possible to see individual strands (2) Camera is far from hair, making details difficult to see. ........................................... 7

1.5 Long, curly, red hair blowing in the wind simulated with level-of-detail techniques.............................................................. 10

1.6 Close camera placement shows high detail on long, curly, red hair blowing in the wind. ................................................................. 11

1.7 Short, blonde hair curled at the bottom (left) dry and (right) wet. ................................. 17

2.1 2D square patch wrapped onto the 3D model by the method of Kim and Neumann [KN02]. ................................................................. 23

2.2 Hair strips as an approximate hair model [KH00]. ....................................................... 25

2.3 The cluster hair model [WY04]. ................................................................. 26

2.4 Modeling hair using a fluid flow [HMT00]. ................................................................. 30

2.5 The polar coordinate system for a hair segment [AUK92]. ........................................ 35

2.6 Hair strand as a rigid multibody serial chain [HMT01]. ........................................... 36

2.7 Fluid Dynamics - Eulerian and Langrangian viewpoints [HMT01]. ............................ 39

2.8 (top) Particles defining hair, line segments indicate direction (bottom) Animation of hair with head shaking [BCN03]. ........................................... 40

2.9 The layered wisp model captures both continuities and discontinuities observed in long hair motion [PCP01]. ........................................... 45

2.10 An electron micrograph of a hair fiber that shows the structure of the outer cuticle surface, which is composed of thin overlapping scales [Rob94]. In this image, the fiber is oriented with the root at the top and the tip at the bottom. . . . 46
2.11 Notation for scattering geometry [MJC+03] ......................... 47
2.12 Comparison between Kajiya's model (left), Marschner's model (middle) and real hair (right). [MJC+03] ................. 51
2.13 Opacity Shadow Maps. (left) Hair volume is uniformly sliced perpendicular to the light direction into a set of planar maps storing alpha values. (right) The resulting shadowed hair [KN01] ................. 56
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 ... 62
3.3 Base skeleton attached to the scalp at the root node (a) Straight resting style (b) Curly resting style .................................................. 65
3.4 Different hairstyles generated using the LOD representations and the base skeleton. From left to right, top to bottom: (1) Short, straight, blonde hair (2) Short, wavy, brown hair (3) Long, curly, blonde hair (4) Horse mane and tail .................. 67
3.5 Cross-section of a strand group and placement of a strand within the group. The skeleton node is at the center of the circular cross-section. The strand placement (shown as the white dot) is determined by its angular placement and percentage of the radius distance value ....................................... 69
3.6 Simulation of long hair blowing in the wind .................................. 71
3.7 Bending motion of a hair skeleton (a) Current positioning of θ, (b) Goal, or rest, position, θo. .................................................. 73
3.8 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 ........................................... 77
3.9 Strip is rendered as a surface but acts like a volume. (top) Cross-sections of three clusters line up with cross-section of strip (shown as brown line); Radius of the clusters used to determine offset of RSS for strip (bottom) Comparison of three clusters and strip with offset, bold black lines indicate volume created for the strip ........................................... 79
3.10 Overlap of two line swept spheres (LSSs). (left) Compute distance d between core lines (right) Subtract offsets to determine overlap value ................... 80
3.11 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 .......................................................... 82
3.12 Shadow effects. Side-by-side comparison (left) without and (center) with hair shadows; (right) Light position and orientation is indicated with grey lines, OSM cutting planes are shown in red around the hair model .................. 86

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
3.13 Visibility of hair is used to choose its representation. (a) User's point of view of the hair (b) Viewpoint from the back to show that hairs the user cannot see are not rendered and are simulated as strips; White lines show the view frustum of the user's camera. ........................................ 91

3.14 Shows quality of each representation to represent a short, curly hairstyle From left to right, top to bottom: (1) LODs (2) Strips (3) Clusters (4) Strands. Note that the visual quality of the LODs is closest to that of the strands. ................................. 95

3.15 Simulation Performance Comparison. Show the factors of speed-up among LOD, strips, and clusters over the strands alone, which is the baseline for comparison. The simulation speed of the system consistently outperforms the strands. It quickly outperforms the use of clusters alone, as the camera starts to zoom out. Then, soon after a certain distance threshold, it performs comparably to the use of strips alone. ................................. 96

3.16 Rendering Performance Comparison. Show the factors of speed-up among LOD (in blue), strips (in green) and clusters (in orange) over the strands alone (in red), which is the baseline for comparison. Notice that the rendering speed of the LOD system consistently outperforms the individual strands. It quickly outperforms the use of clusters alone, as the camera starts to zoom out. Then, soon after a certain distance threshold, it even outperforms the use of strips alone due to the automatic occlusion culling used in the switching criteria for rendering acceleration. ................................. 97

3.17 Comparison simulations of wind blowing through short, wavy hair as the camera zooms out From left to right (1) Strands (2) LOD Representation (3) LOD Representation color coded: strands are shown in yellow, clusters in red, and strips are mostly occluded but shown in blue when rendered. ............ 101

4.1 Hair Hierarchy. One hair hierarchy consists of a single strip hierarchy, multiple cluster hierarchies and multiple strand group hierarchies. The coarsest hair representations are located in the strip hierarchy at the top of the overall hair hierarchy. ........................................ 105

4.2 Strip Subdivision. Strips are split into two equal sized strips. ................. 107

4.3 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 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. ................................. 108

4.4 Strand group hierarchy. Subdivision process creates a quad-tree containing strand group information. Strand group hierarchy can extend to individual strands. ................................. 109
4.5 Adaptive Subdivision: Two skeletons (left) are dynamically subdivided into multiple (right).

4.6 Adaptive Merging. Positional constraints placed on child skeletons merging into parent (a) Parent skeleton (in red) potential position determined by averaging positions 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.

4.7 The rendered images without (left) and with (right) adaptive subdivision.

4.8 A series of snapshots (a)-(f) showing adaptive grouping of hair strands with wind sporadically blowing through long, straight, brown hair.

4.9 Dynamic Simulation of Hair Using LODs. A sequence of snapshots (from left to right).

5.1 Real images of (a) dry and (b) wet hair. Note the difference in the color and hair volume as the strands of hair clump together when wet.

5.2 Real images of hair (a) without and (b) with styling products (hairspray). Note how the hair strands cling to each other, decreasing the overall volume of the hair.

5.3 A curly section of hair can undergo the same global movement, but have differing local motions (a) tight curl is retained (b) loose curl is elongated under force.

5.4 Positioning of the local-skeleton relative to the global-skeleton.

5.5 Localized placement of a line swept sphere. Note that the local placement (shown in red, top) provides a much tighter fit in comparison to previous techniques that place a bounding volume based on a single-skeleton (shown in grey, bottom). (Right) illustrates an entire group of strands with multiple SSVs.

5.6 Sections of curly hair progressively getting wet: (a) 0% wetness (dry) (b) 50% wetness (c) 100% wetness.

5.7 Thin film of water forms on the surface of hair fibers. A layer of water on the fiber surface causes less light to be reflected due to total internal reflection; Water smooths normally rough, tiled hair surface.

5.8 Effects of Water and Styling Products on Hair. Side-by-side comparison of the same hairstyle and color (from left to right) (1) wet (2) normal, dry (3) styling products, note the tighter curl caused by the styling products in comparison to the dry hair.
5.9 Effects of Water and Styling Products on Hair with Motion. Side-by-side comparison of the same hairstyle and color with wind blowing through the hair; TOP: (from left to right) (1) wet hair (2) hair with hairspray; BOTTOM: normal, dry hair. ......................................................... 143

5.10 Dry vs. Wet Hair with Motion. Side-by-side comparison of the same hairstyle and color (top) dry and (bottom) wet. (From left to right) Wind blows through hair and dies down allowing the hair to fall back to rest ........................................ 144

6.1 Hairstyle interactively created with virtual hair salon system. .............. 148

6.2 User Interface PHANToM provides 3D user input and 2D menu buttons are labeled with icons to show applications .......................................... 150

6.3 Simulation Localization (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 .................................................. 152

6.4 (a) Open blades of scissors define cutting triangle (shown in purple) (b) and (c) Grey skeleton (top) remains attached to scalp, red skeleton (bottom) falls down after cut ........................................................... 155

6.5 Example of haircutting, far right shows final style ............................... 156

6.6 (a) User grabs and pulls a section of hair (b) User releases grip and hair falls back to place ................................................................. 158

6.7 Performance comparison between hair wisps and LODs coupled with simulation localization. Rendering consists of shadow and lighting computations on the GPU. Render update includes subdivision of curves and surfaces as well as passing render information to the GPU. Simulation includes implicit integration for dynamics, LOD selection, collision detection (both hair-hair and hair-object), and application processing (blow-dryer, wetting, etc.). The simulation component is accelerated the most in the virtual hair salon, primarily due to the simulation localization feature. .......................... 162

6.8 Simulation Performance Comparison. Shows the factor of speed-up for LODs with simulation 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 .............................................. 163

6.9 Simulation Breakdown. Illustrates the percentage breakdown of each simulation factor using LODs with simulation localization. Note hair-hair and hair-object collision detection still makes-up the majority of the simulation. . . 164

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
6.10 Comparison between real (left) and virtual (right) 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.
Chapter 1

Introduction

Modeling and animating human characters has become an important and active area of research in computer graphics. Virtual humans are used in a variety of applications, making their creation a valuable process. Hair simulation is a vital feature for creating believable virtual humans, but poses many challenging problems mainly due to the high complexity of hair. The human head can possess between 100,000 and 150,000 strands of hair, where the thickness of each strand is quite small, 58 to 100 $\mu$m in diameter [L’05]. Long hair increases this complexity as the number of geometric primitives required to model the strands increases, as does the intricacy of many simulation issues including collisions and hair clustering effects. Figure 1.1 shows differing hairstyles and the complexity involved in real hair.

Due to the overwhelming complexity of hair, typical hair modeling schemes either offer methods to attain high visual quality or high performance speed of the animated hair. Many interactive applications ignore complex features of hair in order to attain a desired performance. By simplifying the hair volume, the visual quality of the hair is often compromised. Meanwhile, in reality, an entire hair volume is typically not fully visible to the viewer. Many hair strands can be occluded by the human avatar or the remaining hair volume. Figure 1.1 illustrates this observation as parts of the hair volume are occluded in each image. Moreover, the placement of the camera and the motion of the hair can dictate where the user’s attention will be focused. Often the desired visual quality of animated hair varies throughout a simulation. When the camera is close to the hair, making the hair strands the primary focus of the viewer, a highly detailed simulation may be desired. Conversely, when the camera is distant
from the hair, a general appreciation for the overall hair volume would usually be sufficient.

There is often a substantial amount of wasted computation allocated towards hairs the viewer hardly notices. However, it can be difficult to determine prior to a simulation which parts of the hair will be important to the application and which parts do not need to be simulated as accurately. Moreover, the focus of the user can change often throughout the simulation, frequently moving these high and low areas of interest.

In this thesis, I propose new methods to determine on the fly which hairs are of most significance to the simulation and novel techniques to allocate the majority of the computational resources towards modeling these hairs. This process then accelerates the simulation of the hairs deemed less important to the simulation, thereby accelerating the overall hair simulation while maintaining the desired visual quality of the total simulated hair.

1.1 Motivation

In this section, I explain the motivations for modeling hair using levels of detail by providing an overview of the problems facing hair modeling and the main applications that benefit from this method.
1.1.1 Problem Overview

Hair modeling can be categorized into three general topics [MTH00]: hairstyling, hair simulation and hair rendering. Hairstyling controls the shape of the hair, while its motion is generated through hair simulation. Hair rendering is responsible for all matters related to depicting the hair visually to the screen. Each of these categories poses unique challenges, but they are still intimately related to each other and complicated by the intricate nature of the hair volume.

Chapter 2 gives a thorough explanation of the challenges facing hair modeling in each of these categories as well as the techniques of previous work to combat these problems. Here I will give a brief overview of the problems facing hair modeling today and how they motivate the use for a level-of-detail framework to model hair.

Specifying the shape of a hairstyle has been accomplished primarily in one of two ways. Users either directly position the geometry of the hair strands to a desired shape or complex physical equations are used to create a general shape for the hair volume. As can be expected, positioning the shape for over 100,000 strands of hair is quite an overwhelming task for an individual to undertake. Even through the recent use of a multi-resolution control structure developed by Kim and Neumann [KN02] that modeled only 5,000 strands on a head, it still took a few hours for a user to position the hair geometry for certain hairstyles. This method, however, was the first to show that a multi-resolution technique could be employed for hair modeling as it was successful in creating numerous intricate hairstyles faster than previous methods had allowed.

On the other hand, volume-based hairstyling techniques require users to set up intricate mathematical formulas to define a shape for the hair volume, such as through fluid flow [HMT00]. It is difficult for many of these techniques to add fine detail to localized regions, such as the creation of a braid, through limitations of the volume. Other methods use a hybrid approach through the use of automatic and manual parameters to add fine detail through additional user defined constraints, which are able to create convincing final results, but the manual time for the complex setup still leads to cumbersome user interactions [CK05].
Given these different approaches, there is currently no technique that allows for hairstyle creation in a manner similar to hairstyle creation in the real world. Natural hairstyling in salons, for example, entails a human stylist interacts with dynamic hair, where the hair is subject to motion from forces created by gravity or the stylist. Modeling this type of interaction would require interactive hair simulation and rendering for an authentic user experience. Due to the limitations of current simulation techniques, however, this type of styling method has not been introduced.

Hair simulation has been one of the most difficult aspects for modeling hair, primarily due to the intricate collisions caused by the high number of deforming hair strands. Given that the thickness of each strand is typically less than 100\(\mu\)m and there are generally over 100,000 strands of hair on a human head, accurately detecting collisions among all the strands is extremely difficult, if not impossible. Detecting collisions between only a fraction of the total strands and the human avatar is similarly quite cumbersome [AUK92].

Following the observation that strands of hair in close proximity with each other move similarly, wisp-based approaches have become popular hair modeling techniques. A wisp is created by grouping numerous individual hair strands into large bundles and each bundle is simulated as separate entities [WS92, KAT93, DTKT93, PCP01, BKC03, CCK05]. The
main complexity to simulating hair wisps is still in handling collisions between the hair and body and especially among the hairs [PCP01]. It is common for these approaches to take several seconds per frame for animation calculations alone. Moreover, wisp methods are limited in that each and every hair strand is still rendered to the screen, creating a high rendering cost.

Real-time hair animation techniques have been used for applications that demand faster processing times than wisps can generate; however these methods significantly ignore many complex hair behaviors, such as mutual hair interactions, dynamic hair clustering effects, or are limited to only simulating short, straight hair. A number of common real-time hair simulation methods approximate the complex geometry of hair with two-dimensional surfaces [KH00, KN00, KH01, GZ02, LH03, SYAM05, TG05]. The visual quality of such methods falls far short of the wisp, or other strand-based techniques due to the flat two-dimensional hair appearance. Figure 1.2 illustrates the visual faithfulness of strips versus wisps for modeling hair.

Hair rendering challenges are similarly limited in that they either offer high visual quality or high performance speed. The thin, semi-transparent nature of hair strands creates quite intricate self-shadowing and lighting effects. Methods that provide the highest visual faithfulness towards real hair model each strand using a high number of line segments (>100K) to create smooth curves. Shadowing effects caused by hair self-occlusions constitute the majority of rendering computation time. Recent interactive hair rendering typically display only a small fraction of strands in comparison to a real hair volume. Using an efficient CPU rendering algorithm, 3,350 hair strands with 100K total line segments can be rendered at around 10 frames per second [BMC05], while recent graphics hardware algorithms have rendered 200K segments at 3 frames per second [MKBR04]. In either method the number of hair strands is the limiting factor for rendering. Again, methods that use surface approximations for the hairs can accelerate hair rendering, but these methods lead to a compromised visual depiction of the hair.

Hair modeling methods are more comprehensively explored in Chapter 2 however, at this time, it is important to note that current methods offer either a high visual quality or a high
performance speed of simulated hair. There are no known techniques that can dynamically balance between the visual fidelity and the performance speed of the hair model. Figure 1.3 illustrates two different methods for modeling hair. The top image shows a still frame from the 2001 computer animated movie “Final Fantasy: The Spirits Within”, though the hairstyle is quite simple (short, straight, black hair) the resulting quality is still superior to current videogame capabilities, an example of which is shown in the bottom image of Figure 1.3 with a screenshot from the 2005 game “The Legend of Zelda: Twilight Princess.” Feature films have strict visual fidelity demands but can perform simulation and rendering computations offline. To achieve these high quality images, hours of computations are needed for simulation and rendering for only a few frames of footage. Meanwhile, videogames have strict performance requirements which have an obvious impact on the visual quality of the animated hair. Given
Figure 1.4: Simulation factors can decide how much detail is visible. From left to right (1) Camera is close to hair making it possible to see individual strands (2) Camera is far from hair, making details difficult to see.

these two widely different examples, there is no method that can switch between the two different techniques on-the-fly throughout a simulation.

Moreover, there is no method that considers the amount of detail the viewer can actually observe to decide how the hair should be modeled. As hair is obviously quite complex, there are often simulation scenarios when the viewer cannot observe a great amount of the hair’s intricate detail. A large portion of the hair can be occluded by the avatar or other objects in the scene or the camera can be located at a large distance from the hair model so that the volume of the hair only covers a small number of pixels on the screen, Figure 1.4 illustrates this example. In these examples, the viewer would not be able to identify each of the hair strands constituting the hair volume; as a result, the use of a coarse hair model would be valid.

Simulation factors have a tendency to change frequently as the camera changes position throughout the scene or as the human avatar moves. When hair lies still, a highly detailed simulation is generally not needed to capture the hair’s behavior; however, when the avatar moves or wind blows through the hair, the motion of the hair can become quite complex desiring a highly detailed simulation. As a result of these changes, the amount of detail desired can vary dynamically over the course of a simulation. Furthermore, the areas within the hair volume that are most prevalent to the viewer also frequently change over time.
These factors motivate the use of levels-of-detail for modeling hair. Through using level-of-detail (LOD) techniques, I will show that the resolution of the simulation can be dynamically changed creating a balance between the visual fidelity and the performance speed of the animated hair. The LOD hair framework will lead to accelerated hair simulation, rendering and styling by allocating the majority of the computational resources towards hairs that are the most significant to the simulation and the viewer.

1.1.2 Applications and Benefits

The applications of hair modeling are quite vast, ranging from the entertainment industry through features films and videogames, to more technical applications such as cosmetic prototyping and training. Applications that stand to benefit the most from a more efficient level-of-detail hair modeling framework are interactive applications that desire higher quality hair simulations than they can currently achieve due to strict performance requirements. Virtual environments frequently depict human avatars and demand strict performance requirements to create convincing interactivity for its users [PSO05]. Flat surfaces are the most common method for modeling hair in these applications. The visual quality of hair animation in these virtual environments can be advanced by increasing the detail simulated in the areas most visible throughout the application.

Similarly, videogames popularly use virtual humans as their central characters. In these games, it is quite common to illustrate the human characters jumping, running or undergoing other high motions that typically create intricate hair dynamics. The videogame characters often have short hair with low to no motion, see Figure 1.3 (bottom). Real-time hair for videogames are typically modeled as flat polygonal surfaces with no shadowing effects [Sch04]. Meanwhile the desire to advance this technology is evident. In a recent article from The New York Times (published May 16, 2005) [Tau05], Glenn Entis, a vice president and the chief visual officer for Electronic Arts stated that Electronic Arts (a top game development company) is looking to create more realistic characters by developing hair that flows more naturally saying that “Hair is such a communicator of style.” Levels of detail offer a way to elevate the visual quality of the animated hair without losing sight of the performance
constraints.

Interactive hairstyle generation is also widely used in multiple areas. Feature films, for example, devote hundreds of man hours towards the creation of believable hairstyles for their characters. Furthermore, interactive hairstyling techniques provide methods for cosmetic stylists to use technology to learn their craft. The cosmetic industry generates billions of dollars in the creation of hair care products. Technological advances in the industry produce another avenue for customer care [L’O05]. As technology advances, the creation of a virtual hair salon could provide a number of advantages. It could appeal to individual users who could customize the system to their own hair or to large corporations for virtually testing cosmetic products. As the technology relating to modeling hair advances, so do the possibilities for its uses.

1.2 Thesis

My thesis is:

*Levels-of-detail can accelerate hair modeling, including styling, simulation and rendering, and provide a balance between computational performance and visual fidelity of animated hair.*

In support of this thesis, I present a level-of-detail framework for modeling hair that incorporates methods for simulating hair dynamics, collision detection, light scattering and self-shadowing effects, as well as effects created by water and styling products on hair. These features together are able to accelerate hair modeling in comparison to traditional strand-based methods and are used to create an interactive hairstyling tool that allows users to interact with dynamic hair through several common hair salon applications. By providing a balance between the performance speed and visual fidelity, a more efficient hair modeling framework is possible.
1.3 Main Results

In this section, I discuss the key results my level-of-detail hair modeling framework offers. I present several advances in hair modeling that help to create an efficient scheme for hair animation. I classify them in the areas of level-of-detail techniques for hair modeling, accelerated hair modeling algorithms, dynamically changing hair properties, and user interaction with dynamic hair. Figures 1.5 and 1.6 illustrate examples of hair simulation my framework is able to capture.

![Image](image.png)

Figure 1.5: Long, curly, red hair blowing in the wind simulated with level-of-detail techniques.

1.3.1 Level-of-Detail Techniques for Hair Modeling

As explained earlier, the complex nature of the hair volume makes simulation, rendering and styling difficult to handle with the high number of deforming hair strands. This observation motivates the use of levels-of-detail for modeling hair, selecting a low level of detail to model hair that is less important to the simulation and the viewer thus drastically reducing computational costs. In this dissertation I present the following key results for the
level-of-detail hair framework:

- Three discrete representations for modeling hair: *individual strands, clusters* and *strips.*
  Each representation provides a different level of visual fidelity and performance speed. (Chapter 3)

- Dynamic tessellation of each representation through the use of subdivision curves and surfaces. (Chapter 3.3)

- A *hair hierarchy* structure that creates varying resolutions of each discrete representation to provide more localized control over the modeling of hair and to capture dynamic grouping and splitting effects of hair. (Chapter 4)

- Criteria for determining the most significant sections of hair to the viewer and application. (Chapter 3.6 and 6.2)

- Transitioning methods for changing the LOD representation and resolution on-the-fly as the simulation changes. (Chapter 4.2 and 6.4)
The level-of-detail framework that I present for modeling hair automatically simplifies the simulation and rendering of hairs that are of low importance to the viewer and application. This framework is built from the creation of three discrete representations for modeling hair. The individual strands provide the highest level of visual fidelity for representing hair but require the most computation, whereas the strips provide the fastest computation but the lowest level of visual fidelity. The cluster representation falls between the strands and the strips for both computation speed and visual faithfulness towards modeling hair. These LOD representations are used for continuous level-of-detail rendering through adaptive tessellation of the subdivision curves and surfaces that define their geometric construction.

Along with the discrete LOD representations for hair, I present the hair hierarchy, a structure that is built from the continual subdivision of the discrete representations. The hair hierarchy, pre-computed offline, offers further refinement and control over the simulated detail as it contains varying resolutions of each discrete representation. The hair hierarchy is traversed on-the-fly throughout a simulation to find the appropriate resolution for a section of hair. Moreover, the hair hierarchy captures an additional hair behavior: it can model adaptive grouping and splitting behaviors exhibited in real hair. Due to static electricity, oils on the hair or styling products, strands in close proximity tend to follow similar motion. Under large motions, real hair exhibits adaptive clustering effects as large groups of strands split into smaller groups or hair strands move from one group to another. The hair hierarchy automatically captures these adaptive clustering effects, a difficult, if not impossible, achievement for many other hair modeling schemes.

I have identified several criteria for sufficiently determining the significance of a section of hair to the viewer and the simulation. These criteria judge the amount of detail the viewer can recognize for a given hair section based on the hair’s visibility, viewing distance, and motion. Chapter 3 will explain these criteria and the methods I have developed for measuring them. Chapter 6 introduces the user's direct interaction with hair as an extra criterion for determining the appropriate simulation resolution and describes a simulation localization technique based on these findings. As the simulation changes over time, moving the areas of highest importance, these criteria are continually used to update the status of
each section of hair.

The changing status of a hair section signifies that a level-of-detail transition is to take place. Transitions between LODs must be smooth so that they do not distract from the visual quality of the simulation. I have developed several methods to create a smooth transition from one level of detail to the next. Each LOD representation uses the same underlying control structure that dictates hair placement and motion, which simplifies property inheritance during a transition. Rendering techniques are also used to alleviate visual distractions that can occur during LOD transitions. Moreover, I impose a number of transition constraints that help to assure LOD transitions will only occur when they will be least noticed by the viewer.

I have successfully used the criteria that are detailed in Chapters 3 and 6 to dynamically determine areas of the hair that make the most significant impact to the viewer and the application and have been able to change the level of detail of the hair on the fly as the simulation proceeds. I have tested and demonstrated this level-of-detail framework through several simulations to judge the impact of the transitions to the user and have compared these results with similar simulations that do not use levels of detail to judge the visual fidelity of the overall simulated hair. The comparisons have shown that the visual faithfulness of hair modeled with LOD techniques remain comparable to the use of strands alone. Moreover, through using hair LODs the dynamic simulation can be accelerated up to two orders of magnitude and the rendering performance accelerated up to six times in comparison to the use of strands alone.

1.3.2 Accelerated Hair Modeling Algorithms

I have extended the basic components of hair simulation and rendering from previous research methods, however there are multiple procedures in hair modeling that are highly complex due to the great number of strands on the human head. The level-of-detail framework aids in automatically simplifying these tasks, but I have developed several accelerated algorithms that have been incorporated into the framework that provide further computational efficiency.
In Chapter 3, I explain the following algorithms and their incorporation into the level-of-detail framework:

- Dynamic simulation with an implicit integration scheme designed for characteristic hair motion to provide greater stability. (Chapter 3.4.1)

- Collision detection built from tight-fitting bounding volumes for each of the LOD hair representations, which provides a simple and fast intersection test. (Chapter 3.4.2)

- A method to capture the multiple modes of light scattering from hair using graphics hardware acceleration. (Chapter 3.5.1)

- Shadowing effects due to hair self-occlusions and avatar occlusion using recent GPU features. (Chapter 3.5.2)

I propose the use of implicit integration for hair simulation since it offers advantageous stability properties in comparison to traditional explicit integration techniques such as Euler's or fourth-order Runge-Kutta [BW98]. Implicit integration allows for the use of stiffer springs and larger time steps, which help to model a wider variety of hair types and hair motions over previous techniques. For example, the application of hairspray increases the stiffness of hair fibers noticeably and through the use of implicit integration, these changes can be modeled with no effect on the time step being used.

Collision detection, as explained earlier, has been identified in many hair modeling applications as the most time consuming process related to capturing hair motion. I propose the use of the family of swept sphere volumes [LGLM00] as novel bounding volumes for hair. I will show how these bounding volumes fit tightly around the hair geometry and provide a simple intersection test. Furthermore, I have successfully used these bounding volumes to detect collisions between the hair and the avatar as well as to handle complex hair mutual interactions. My collision detection and response methods are demonstrated on various simulation scenarios, including the intricate case when twisted sections of hair are allowed to unravel around each other.
I have also integrated various hair rendering techniques that utilize recent graphics hardware advances. By performing these algorithms using graphics hardware I will show that convincing shadowing effects and multiple modes of light scattering from hair fibers can be calculated. I will show the effectiveness of these algorithms by comparing the visual quality of hair with and without shadowing and light scattering.

Each of these techniques is able to accelerate certain basic features of hair modeling. Furthermore, they can function independently from the LOD framework for general hair modeling. The achievements they offer to animated hair are not strictly limited to levels of detail for hair, but they strengthen the LOD framework by offering efficient algorithms for essential computations.

1.3.3 Dynamically Changing Hair Properties

I present multiple methods for dynamically changing the properties of hair in the presence of external substances such as water and styling products. The cosmetic industry offers a myriad of styling products that alter the behavior of hair. Additionally, water is commonly applied to hair through activities such as washing or swimming and greatly changes the physical and visual properties of hair. Modeling the basic behaviors of hair is a challenging task, but capturing the changes created from these common external substances on hair offers a broader generality and applicability of the hair model.

In Chapter 5 I introduce the following mechanisms:

- A dual-skeleton system that uncouples hair properties so they can be altered separately from external substances. (Chapter 5.2)

- Automatic adjustment of dynamic properties due to water or styling products. (Chapter 5.3.1)

- The design of flexible geometric structures that account for frequently changing hair volume. (Chapter 5.3.2)

- Dynamic bonds that model adhesive forces introduced by styling aids on the hair.
(Chapter 5.3.3)

- Parameterization of the key factors for lighting and shadowing equations that affect the visual appearance of wet hair. (Chapter 5.4)

The behavior and appearance of hair are largely influenced by the substances present on it. I use knowledge from biologic and chemical literature concerning the influence of water and styling products on hair to create the first algorithms for effectively modeling these results on hair.

I introduce a dual-skeleton structure for modeling hair that is able to capture many of the changing hair properties encountered. The dual-skeleton is the first proposed structure that can separate the control over several attributes of hair. Since water and styling products influence these attributes in different ways, the dual-skeleton allows for each attribute to be altered independently as water or styling products are applied to the hair. The dual-skeleton can capture changes in mass and spring stiffness that affect the hair's dynamic motion. Moreover, dynamic curling and straightening of hair is modeled through the dual-skeleton, a property largely influenced by cosmetic styling aids. Additionally, I propose a technique, referred to as dynamic bonds, which effectively capture the adhesion created between sections of hair when fixative products are applied to the hair.

I also present a flexible geometric structure that controls the volume of the hair, dynamically adjusting it when water or styling products are used. The changing volume influences the dynamic motion and collision detection then associated with the hair; this adaptability is particularly important to model the dynamic application of water onto hair.

The interaction of light with an object is largely changed as the object becomes wet, as observed by [JLD99]. I extend the methods first proposed by [JLD99] to render wet hair by identifying and parameterizing the key rendering factors influenced by water for hair rendering. The results recreate the appearance of wet hair by adjusting the shadowing and light scattering computations on hair.

I have effectively captured these changing properties associated with the motion and visual depiction of hair influenced by water and styling products. I illustrate the results in multiple
simulations that show hair as water is dynamically applied to it, changing the motion and rendering properties of the hair on the fly, as well as through side-by-side comparisons of wet and dry hair, refer to the example in Figure 1.7. Furthermore, I compare the simulated results with real images of wet and dry hair to highlight how the key properties of wet hair have been captured. Similarly, to illustrate the effectiveness of modeling styling products on hair I show comparison simulations of hair with and without styling products, which in particular highlight the adhesiveness of strands and retention of curls when styling products are applied. I also show the effects of hairspray on real hair to validate these observations.

1.3.4 User Interaction with Dynamic Hair

This dissertation covers a wide range of areas in hair modeling. The level-of-detail framework is able to identify areas of the hair that are of highest importance to the simulation and allocate the majority of the computational resources towards modeling those hairs. I also present numerous methods for accelerating essential hair algorithms and methods for dynamically capturing the influences of water and styling products on hair. Along with the results described earlier, I also illustrate these contributions together for the purpose of interactive hairstyling, which is described in Chapter 6. I present a way for the user to directly change the properties of hair on the fly through several common hair salon applications with direct
user manipulation. Specifically, in Chapter 6 I present:

- A user interface that attains 3D input from the user for direct manipulation of the hair. (Chapter 6.1)

- Simulation localization techniques that focus the simulation of the hair based on user interaction allowing for the user to directly interact with dynamic hair. (Chapter 6.2)

- Several common hair salon applications that alter the look, structure, or physical behavior of hair (e.g. hair cutting, wetting, blow-drying). (Chapter 6.3)

By using levels-of-detail to model hair, I create a physically-based virtual hairstyling system that mimics real-world hairstyling processes more intuitively than previous methods have allowed. The techniques I present for interactive hairstyling allow for the user to directly manipulate dynamic hair through several common hair salon applications. The user requires no knowledge about hair modeling other than common hair manipulation techniques. With an intuitive 3D interface, users can directly manipulate and position hair strands, as well as employ real-world styling applications (e.g. cutting, wetting, applying styling products) to create hairstyles in a manner similar to the same processes in the physical world.

I demonstrate the usefulness of the interactive hairstyling system by showing several hairstyles created by direct user manipulation. I also compare an assortment of the supported applications to the same processes in the real world to show the system's ability to dynamically change the properties of hair at local areas on the hair model in a manner faithful to real world hairstyling.

1.4 Thesis Organization

The rest of this dissertation is organized as follows. The next chapter summarizes related work in hair modeling and model simplification. The level-of-detail framework for modeling hair and its results are described in Chapter 3. The construction and benefits of the hair hierarchy as well as LOD transition details are illustrated in Chapter 4. Chapter 5 then
discusses the methods for capturing the effects of water and styling products on hair. Interactive hairstyling using direct user manipulation of hair is explained in Chapter 6. Finally in Chapter 7, I summarize my results for level-of-detail hair modeling and conclude with future possible research directions.
Chapter 2

Related Work

Modeling hair involves hair shape modeling, dynamic hair simulation, and hair rendering. The relevant work to this research include the areas of modeling hair as well as geometric and simulation level of detail (LOD) techniques.

As illustrated by Magnenat-Thalmann et al. [MTH00], hair modeling can be divided into three general categories: hairstyling, hair simulation, and hair rendering. Hairstyling, which can be viewed as modeling the shape of the hair, incorporates the geometry of the hair and specifies the density, distribution, and orientation of hair strands. Hair simulation involves the dynamic motion of hair, including collision detection between the hair and objects, such as the head or body, as well as hair mutual interactions. Finally, hair rendering entails color, shadows, light scattering effects, transparency, and anti-aliasing issues related to the visual depiction of hair on the screen. ( Portions of the text from this chapter appear in the recent survey on hair modeling by Ward et al. [WBK+05].)

2.1 Hairstyling

Hairstyling can be a complex process broken into several key steps. Hair strands need to be placed onto the head of an avatar in a desired position, distribution, and orientation. The shape of the hair strands needs to be specified, whether they are long or short, straight or curly, etc. More intricate hairstyles can involve braids, ponytails, or clips in the hair that will require gathering sections of hair to follow certain orientations and given positional
constraints.

Creating a desired hairstyle can often be a long, tedious, and non-intuitive process. Numerous hairstyling techniques have been presented possessing various user interaction requirements. In this section, previous work for styling hair will be presented.

2.1.1 Hair Placement on Scalp

The first step to modeling human hair is simply placing the hair on the avatar's scalp. However, this task is not so simple in practice due to the high number of hair strands composing a human head of hair, making it seem impossible to manually place each hair strand onto the scalp. To simplify the process, a number of intuitive techniques have been developed that employ 2D or 3D placement of hairs onto the scalp as well as methods to control distribution of hair strands around the scalp.

2D Strand Placement

In some styling approaches, hair strands are not directly placed onto the surface of the head model. Instead, the user interactively paints hair locations on a 2D map, which is subsequently projected onto the 3D model using a mapping function. In the method presented by Rosenblum et al. [RCT91], the user edits hair roots on a 2D follicle map; the editor provides symmetry operations so that the user only has to specify the map for one side of the object. Spherical mappings to map the strand bases to the 3D contour of the scalp have been popular approaches [RCT91, Yu01].

Alternatively, Kim et al. [KN02] define a 2D parametric patch that the user wraps over the head model, illustrated in Figure 2.1. The user can interactively specify each control point of the spline patch. In the 2D space defined by the two parametric coordinates of the patch, the user can place various cluster sections.

Placing hair roots on 2D geometry is easy for the user and allows flexibility. But mapping 2D hair roots onto a 3D curved scalp may cause distortion. Bando et al. [BCN03] use a harmonic mapping and compensate for the mapping distortion by distributing the root particles based on a Poisson disc distribution using the distance between corresponding points.
Figure 2.1: 2D square patch wrapped onto the 3D model by the method of Kim and Neumann [KN02].

on the scalp in world space rather than their 2D map positions.

3D Strand Placement

Alternatively, hair roots can be directly placed on the scalp using 3D placement. Patrick et al. [PB03] present an interactive interface where the user can select triangles of the head model. The set of selected triangles defines the scalp, i.e. the region of the head mesh where hair will be attached. A wisp is created for each triangle of the scalp. Shapes of the resulting wisps thus strongly depend on the head mesh, which could be awkward if some control on the wisps' size is desired.

Distribution of Hair Strands on the Scalp

Typically, hair strand roots are uniformly distributed over the scalp since this is a good approximation of real hair distribution. Since hairs in local proximity to each other on the scalp tend to group and move together throughout the length of the strands due to static electricity or oils on the hair, hair *wisps* have been created to group strands of hair into bundles. A hair wisp is typically defined by a general 3D curve in space and a contour shape, such as a circle or triangle, which surrounds the curve, defining the shape of the wisp. Rendered strands are then placed inside of the wisp. Some wisp-based approaches randomly distribute hair roots inside the region of the scalp covered by the root wisp section [PCP01, CK05, CSDI99]. But if wisp sections overlap, a higher hair density is generated in the overlapping regions, which can produce distracting results. In order to guarantee a
uniform hair distribution over the whole scalp, Kim et al. [KN02] uniformly distribute hair over the scalp and then assign each generated hair root to its owner cluster.

Some approaches also enable the user to paint local hair density over the scalp [HR04, CK05]. Hair density can be visualized in 3D by representing density values as color levels. Controlling this parameter is helpful to produce further hairstyles such as thinning hair. Hernandez and Rudomin [HR04] extended the painting interface to control further hair characteristics such as length or curliness.

2.1.2 Geometry-Based Hairstyling

Geometric-based hairstyling approaches mostly rely on a parametric representation of hair in order to offer an intuitive and easy-to-use interface to the user. These parametric representations can involve surfaces to represent hair or wisps in the form of trigonal prisms or generalized cylinders. The manual editing of such primitives often produces a global shape for hair.

Parametric Surface

Using two-dimensional surfaces to represent groups of strands has become a common approach to modeling hair [KH00, LH03, NT04]. Typically in these methods, using a patch of a parametric surface, such as a NURBS surface, reduces the number of geometric objects used to model a section of hair and helps accelerate simulation and rendering. These NURBS surfaces, often referred to as hair strips, are given a location on the scalp, an orientation, and weighting for knots to define a desired hairstyle. Texture mapping and alpha mapping are then used to make the strip look more like strands of hair. A complete hairstyle can be created by specifying a few control curves, or hair strands. The control points of these hair strands are then connected horizontally and vertically to create a strip. Though this method can be used for fast hairstyle generation and simulation, the types of hairstyles that can be modeled are limited due to the flat representation of the strip (see Figure 2.2).

In order to reduce this flat appearance of hair, Liang and Huang [LH03] use three polygon meshes to warp a 2D strip into a U-shape, which gives more volume to the hair. In this
method, each vertex of the 2D strip is projected onto the scalp and the vertex was then connected to its projection.

Kim and Neumann [KN00] developed a method to give a realistic appearance to hair models that are represented as a set of parametric surfaces. The *Thin Shell Volume*, or TSV, starts from a parameterized surface representation of the hair. Thickness is added to the hair by offsetting the surface along its normal direction. Individual hair strands are then distributed inside the TSV.

NURBS surfaces are also used to create a volume of hair by Noble and Tang [NT04] in the case of cartoon hair. Here, a NURBS volume is created and shaped to a desired hairstyle. Key hair curves are generated along the isocurves of the NURBS volume, following the motion of the original surface. Extra geometric detail is added to the hair by profile curves that are extruded from the key hair curves creating extra clumps of hair. These clumps of hair can then be animated independently from the original NURBS surface.
Trigonal prism

Wisps of hair strands have been created in the form of trigonal prism wisps that reduce the number of control parameters needed to define a hairstyle [WS92, CSDI99]. Chen et al. [CSDI99] create a wisp using a boundary of three 3D B-spline curves that can easily be edited to change the shape of the wisp. To draw hair strands inside a wisp, 2D coordinates are randomly chosen inside a 2D circular section for each hair strand; these coordinates are used as barycentric coordinates inside the circumscribed circle around each triangular wisp section to define the location of the points interpolating each hair strand.

Generalized Cylinder

Similar to the use of wisps for modeling hair, generalized cylinders, or clusters, have been used to create hairstyles [YXWY00, XY01]. The cluster hair model is created from hair strands distributed inside of a generalized cylinder, see Figure 2.3. To define a cluster, a general space curve is created and used as the center of a radius function that defines the cross-section of the cluster.

Figure 2.3: The cluster hair model [WY04]

V-HairStudio, a tool created by Xu and Yang [XY01], allows a user to produce and manipulate hair clusters to generate a complex hairstyle. Users employ 3D interaction to
manipulate the cluster axis curve, positioning the hair strands in desired locations and shapes. Intermediate results can be viewed quickly by previewing only the cluster-axes to show basic positioning of the hair, or the clusters can be shown as surfaces to view the spatial relationship between hair clusters and the head model.

A similar technique has been used to control braids and twists to model African hairstyles [PBL04]. The generalized cylinders allow for many popular hairstyles that involve constrained shapes such as braids or ponytails. Some more complex hairstyles that do not rely on strands grouped into fixed sets of clusters are more difficult to achieve with this method.

**Multi-resolution Editing**

The principle of representing global hair shape with generalized cylinders can be further extended to provide multi-resolution control in hair shape editing. Kim and Neumann [KN02] represent complex hair geometry with a hierarchy of generalized cylinders, allowing users to select a desired level of control in shape modeling. Higher level clusters provide efficient means for rapid global shape editing, while lower level cluster manipulation allows direct control of detailed hair geometry, down to every hair strand. Complex hairstyles such as curly clusters can be efficiently generated with a copy-and-paste tool that transfers detailed local geometry of a cluster to other clusters.

Using a similar idea, Wang et al. [WY04] extended the cluster hair model [YXWY00, XY01] to a hierarchical cluster hair model, where the shape of hair can be edited at multiple levels of detail (the group level, the cluster level and the individual hair strand level) from top to bottom.

**Local Shape Editing**

Once a global shape has been defined for hair (using either a geometric-based approach or a physics-based approach, see Section 2.1.3), local details such as curls, waves or noise might need to be added to achieve a natural appearance for hair.

Yu [Yu01] generates different kinds of curliness inside hair using a class of trigonometric offset functions. Various hairstyles can thus be created by controlling different geometric
parameters such as the magnitude, the frequency or the phase of the offset function. In order to prevent hair from looking too uniform, offset parameters are combined with random terms that vary from one hair cluster to another.

Choe et al. [CK05] model a hairstyle with several wisps, and the global shape of each wisp is determined by the shape of a master strand. Within a wisp, the degree of similarity among the strands is controlled by a length distribution, a deviation radius function and a fuzziness value. The geometry of the master strand is decomposed into an outline component and a details component. The details component is built from a prototype strand using a Markov chain process where the degree of similarity between the master strand and the prototype strand can be controlled through a Gibbs distribution. Resulting hairstyles are thus globally consistent while containing fine variations that greatly contribute to their realism.

2.1.3 Physics-based Hairstyling

Some hairstyling techniques are strongly linked to physically-based animation of hair. Section 2.2 gives a thorough explanation of dynamic hair simulation. This section focuses on the use of physics for hairstyle generation, including hair shape specification and hair volume creation. Many of these methods allow for the generation of hairstyles that are controllable through physical parameters.

Hair Shape Generation

While the global shape of hair can be dictated using methods described in Section 2.1.2, the shape can also be created through the specification of a few key parameters using physically-based methods ranging from cantilever beams that control individual strands to fluid flow methods that control the volume of hair.

The cantilever beam In the field of material strengths, a cantilever beam is defined as a straight beam embedded in a fixed support at one end only, the other end being free. Anjyo et al. [AUK92] consider that it is a similar case to a human hair strand, where the strand is anchored at the pore, and the other end is free. Considering gravity is the main source of
bending, the method simulates the statics of a cantilever beam to get the pose of one hair strand at rest. Combining this technique with hair-head collision handling, hair shearing, and the application of additional external forces, results in several different smooth hairstyles.

**Particles** Stam [Sta95] showed that hair could be modeled using particles in motion fields. A particle is given a fixed life-time and traced through a motion field. The history of the particle comprises the whole hair strand. Changing the life-time of the particle then changes the length of the hair. Dynamics can be added to the hair by changing the motion fields over time. Collisions among hair strands and between hair strands and objects are handled automatically through the continuity of the motion field. To create more complex hair shapes the user perturbs each hair individually, such as adding a twist to a hair strand to create a curl.

**Fluid Flow** Hadap and Magnenat-Thalmann [HMT00] modeled static hairstyles as streamlines of fluid flow based on the idea that static hair shapes resemble snapshots of fluid flow around obstacles. In this method, hair-hair collision is avoided through the continuum property of fluid where no two streamlines of fluid flow intersect. Moreover, hair-body interactions are avoided during styling due to the use of fluid flow around objects.

The user creates a hairstyle by specifying certain properties. Streams, vortices and sources are placed around the model to define a hairstyle. A vortex is used to create a curl in the hair at a desired location (see Figure 2.4).

Hadap and Magnenat-Thalmann gave a more natural look to the hair by incorporating a breakaway behavior to individual hair strands that allowed the strand to breakaway from the fluid flow based on a probability function. Otherwise an overall perturbation to the flow field could be defined to alter the general style.

Hadap and Magnenat-Thalmann later extended this work to simulate dynamic hair, as explained in Section 2.2.2.

**Styling Vector Field** Yu [Yu01] observed that both vector fields and hair possess a clear orientation at specific points while both are also volumetric data; this led him to the use of
static 3D vector fields to model hairstyles. This method models global and local hair flow by superimposing procedurally defined vector field primitives. Given the global field then defined by these vector field primitives, hair strands are extracted by tracing the field lines of the vector field. A hair strand begins at a designated location on the scalp and then grows by a certain step size along the direction of the accumulated vector of the vector field. The strand continues to grow until its desired length is reached.

Choe et al. [CK05] also use a vector field to compute global hair position while accounting for hair elasticity. Their algorithm calculates hair joint angles that best account for both the influence of the vector field and the natural trend of the strand for retrieving its rest position. Another important feature of the approach is the ability for the user to define hair constraints. A hair constraint causes a constraint vector field to be generated over a portion of 3D space that modifies the original vector field proportionally to a weight parameter. Hair deformation is computed by using the previous algorithm applied on the modified vector field. In practice, the user can specify three types of constraints: point constraints, trajectory constraints and direction constraints. Hair constraints turn out to be very useful for creating complex hairstyles involving ponytails, bunches or braids.
Producing Hair Volume

Whereas most geometric-based hairstyling methods implicitly give volume to hair by using volumetric primitives (see Section 2.1.2), physically-based methods often account for hair self-collisions in order to produce volumetric hairstyles. Approaches that view hair as a continuous medium [Sta95, HMT00, Yu01] add volume to the hair through the use of continuum properties that reproduce the effects of collisions between hair strands, such as via vector fields or fluid dynamics. As strands of hair become closer, these techniques either prevent them from intersecting due to the layout of the vector or motion fields, or foster a repulsive motion to move them apart from each other, causing the hair to appear thicker.

Since detecting collisions between strands of hair can be difficult and, in the least, very time consuming, Lee and Ko [LK01] developed a technique that adds volume to a hairstyle without locating specific intersections among strands. The idea is that hair strands with pores at higher latitudes on the head cover strands with lower pores. Multiple head hull layers are created of different sizes from the original head geometry. A hair strand is checked against a specific hull based on the location of its pore. A hair-head collision detection and response algorithm is then used. Of course, this method only works in the case of a quasi-static head that remains vertically oriented.

In Choe et al.'s method [CK05], hair density is considered as a measure of collisions; when the density of a hair region reaches a given threshold, a collision is detected and hair is simply forced to occupy a larger volume.

Further hair-hair interaction methods specific to dynamic hair are discussed in Section 2.2.

2.1.4 Generation of Hairstyles from Images

Generating a realistic hairstyle using a modeling interface such as the ones presented in Section 2.1.2 generally takes hours of manual design. Approaches that focus on modeling the whole volume of hair (Section 2.1.3) can help the user in generating the global shape of hair automatically, but fine hair details have to be added using a procedural technique. Using pre-
vious methods can thus be very tedious to model a specific hairstyle. Recent approaches have proposed an alternative way of generating hairstyles based on the automatic reconstruction of hair from images.

**Hair Generation From Photographs**

Kong *et al.* were the first to use real hair pictures to automatically create hairstyles [KTN97]. Their method is merely geometric and consists of building a 3D hair volume from various viewpoints of the subject’s hair. Hair strands are then generated inside this volume using a heuristic that does not ensure faithfulness in hair directionality. This approach is then best suited for simple hairstyles.

Grabli *et al.* introduced a new approach exploiting hair illumination in order to capture hair local orientation from images [GSML02]. Their system works by studying the reflectance of the subject’s hair under various controlled lighting conditions. Fixing the viewpoint allows them to work with perfectly registered images. By considering a single viewpoint and using a single filter to determine the orientation of hair strands, the method reconstructs hair only partially. Paris *et al.* extended this approach [PBS04] to a more accurate one, by considering various viewpoints as well as several oriented filters; their strategy mainly consists of testing several filters on a given 2D location and choosing the one that gives the most reliable results for that location. The method captures local orientations of the visible part of hair, and thus produces visually faithful results with respect to original hairstyles.

Starting from the work by Paris *et al.*, Wei *et al.* [WOQS05] greatly improved the flexibility of the method. In their work, exploiting the geometry constraints inherent to multiple viewpoints proves sufficient to retrieve a hair model, with no need for controlled lighting conditions or a complex setup.

**Hair Generation From Sketches**

Mao *et al.* [MKIA04] developed a sketch-based system dedicated to modeling cartoon hairstyles. Given a 3D head model, the user interactively draws the boundary region on the scalp where hair should be placed. The user then draws a silhouette of the target hairstyle.
around the front view of the head. The system generates a silhouette surface representing the boundary of the hairstyle. Curves representing clusters of hair are generated between the silhouette surface and the scalp. These curves become the spine for polygon strips that represent large portions of hair, similar to the strips used by [KH00, LH03].

This sketch-based system quickly creates a cartoon hairstyle with minimal input from its user. The strips, or cluster polygons, used to represent the hair, however, are not appropriate for modeling more intricate hairstyles such as those observable in the real world.

2.2 Hair Animation

The animation of hair motion involves several components. First, the dynamic motion of hair has to be processed; then, the interaction of hair with other objects, such as the body of the avatar, and hair self-collisions need to be computed. Accurately simulating hair and all of its interactions can be an overwhelming process for many systems, due to the high number of hair strands.

Hair simulation techniques differ widely in performance and resulting visual quality. Additionally, researchers have approached simulating hair in diverse ways. The techniques have been categorized based on various interpretations. Early researchers viewed a full head of hair as individual hair strands that are animated separately. While there can be over 100,000 strands of hair on a human head, it was readily observed that most strands of hair tend to move similar to each other, as if they were a part of a whole continuum. This observation was extended to see hair as large sets of disjoint groups. Each group is animated separately from the others and contains multiple strands of hair.

2.2.1 Animation of Individual Hair Strands

Early hair animation approaches considered each hair strand as an independently moving entity [RCT91, AUK92, RH96], and their main focus consisted of simulating the motion of individual hair strands properly.

A hair strand is an anisotropic deformable object: it can easily bend and sometimes twist
but it strongly resists stretching or shearing. A hair strand also tends to recover its original shape after the stress being applied to it has been removed. Lastly, hair strands are very light and thin objects, and a human head is composed of over 100,000 hair strands. Consequently, one can imagine why most approaches simulating individual hair strands cannot afford to process hair mutual interactions. However, such approaches have the merit of having been significantly reused in further work mainly dealing with simulation of guide hair strands and wisps.

Mass-Spring Systems

One of the first attempts to animate individual hair strands was presented by Rosenblum et al. [RCT91] in 1991. A single hair strand is modeled as a set of particles connected with stiff springs and hinges. Each particle has three degrees of freedom, namely one translation and two angular rotations. This method is simple and easy to implement. However, the fact that a hair strand hardly stretches requires the use of strong spring forces, which leads to stiff equations that often cause numerical instability, unless very small time steps are used. The enormous amount of computation required for the examples presented in [RCT91] limited the number of strands simulated to about 1000 at that time.

One Dimensional Projective Equations

In 1992, Anjyo et al. proposed a simple method based on one-dimensional projective differential equations for simulating the dynamics of individual hair strands. Initially, the statics of a cantilever beam is simulated to get an initial plausible configuration of hair strands. Then, each hair strand is considered as a chain of rigid sticks $s_i$. Hair motion is simulated as follows:

- Each stick $s_i$ is assimilated as a direction, and thus can be parameterized by its polar angles $\phi$ (azimuth) and $\theta$ (zenith) (see Figure 2.5).

- The external force $\mathbf{F}$ applied to the stick is projected onto both planes $P_{\phi}$ and $P_{\theta}$, respectively, defined by $\phi$ and $\theta$ (the longitudinal projection of $\mathbf{F}$ on $s_i$ is neglected
Figure 2.5: The polar coordinate system for a hair segment [AUK92].

since it should have no effect on the rigid stick).

• Fundamental principles of dynamics are applied to each parameter \( \phi \) and \( \theta \) which leads to two differential equations that are solved at each time step. Positions of the sticks representing each hair strand are always processed from the pore to the tip, where the root stick position is first processed and following sticks are recursively defined.

• Collisions between the hair and the head are processed by using a simple ellipsoid-based pseudo force field.

This method is attractive for many reasons: it is easy to implement, efficient (tens of thousands of hair strands can efficiently be simulated this way), and it produces quite realistic motion for hair; this might explain why so many researchers [KAT93, DTKT93, RH96, LK01] have subsequently chosen to use it, especially for animating groups of hair strands (see Section 2.2.3). However, as the method is not based on rigorous physics, it suffers from issues related to collision handling, which are discussed in Section 2.2.1.

Chain of Articulated Bodies

In order to compute the motion of individual hair strands, forward kinematics have been used as a more general alternative to one-dimensional projective equations [HMT01, CJY02].
Such techniques are well-known in the field of robotics, and efficient algorithms for processing dynamics of a multibody open chain have been proposed for quite a long time [Fea87].

Hadap and Magnenat-Thalmann [HMT01] propose a hybrid model for hair to account for both individual hair behavior and complex hair self-interactions. Hair interactions are handled using continuum dynamics, as explained in Section 2.2.2. Individual geometry and stiffness are modeled using the dynamics of an elastic fiber. Each hair strand is represented as a serial rigid multibody open chain using the reduced (or spatial) coordinates formulation [Fea87] in order to keep only the bending and twisting degrees of freedom of the chain: stretching DOFs are removed. Consequently, each hair strand is represented by a set of rigid links connected by three DOFs spherical joints (see Figure 2.6). Apart from the gravitational influence, two kinds of forces are applied on each link. First, a function of joint variables incorporating the bending and torsional stiffness constants, namely a joint actuator force, accounts for the bending and torsional rigidity of the hair strand. Second, a response force (computed by the continuum model, see Section 2.2.2) is applied to account for all of the effects of interactions such as hair-hair collision, hair-body collision and hair-air drag. Forward dynamics is processed using the Articulated-Body Method described in [Fea87], with a linear time complexity.

A similar approach is used by Chang et al. [CJY02] to animate sparse guide hair strands. Kinematics equations are expressed using the generalized coordinates formulation.
Interaction Handling

When simulating hair as a set of individual hair strands, the handling of interactions can be tricky and depends on the method chosen for the strand dynamics, as well as the number of strands simulated.

**Hair-Body Interactions**  In methods simulating chains of rigid links or articulated bodies, motion is processed from top to bottom. This means that a collision detected at stick $s_k$ only affects the following sticks $s_j$, where $j > k$ without propagating the effect backward to the sticks located near the roots, which can lead to unrealistic shapes for hair. In the case of articulated bodies, inverse kinematics could be used to adjust the positions of the whole strand according to the node displacement due to the collision. However, solving the inverse kinematic equations can be costly to compute. Refining the method based on a chain of rigid links, Lee and Ko [LK01] simply fix the problem by adding an extra force that enables hair to get a proper shape when colliding with an object other than the head.

As opposed to these two approaches, the mass-spring simulation ensures straightforward collision handling.

**Hair-Hair Interactions**  Most of the methods simulating individual hair strands neglect hair-hair interactions for efficiency reasons, failing to represent the actual complexity of hair during motion (hair looks very smooth and lacks volume), and do not account for the dissipation of energy due to friction between hair strands.

Lee and Ko [LK01] refined Anjiyo's approach by proposing a very simple algorithm for hair-hair interactions, based on multiple head layers. However, this simple approach is limited to very gentle motion, as hair is assumed to remain close to the head during motion. It is more suited for inserting volume into a hairstyle as explained in Section 2.1.3.

Other approaches for animating hair make assumptions on hair consistency during motion to simplify the problem of collisions. Basically, hair is either globally considered as a continuous medium (Section 2.2.2), or as a set of disjoint groups of hair strands (Section 2.2.3). Specific hair-hair interaction models are proposed in both cases.
2.2.2 Hair as a Continuous Medium

Due to the high number of strands composing a human head of hair, simulating each strand individually is computationally overwhelming. Furthermore, strands of hair in close proximity with each other tend to move similarly. This observation led researchers to view hair as an anisotropic continuous medium.

Animating Hair using Fluid Dynamics

Considering hair as a continuum led Hadap and Magnenat-Thalmann [HMT01] to model the complex interactions of hair using fluid dynamics. The interactions of single hair strands are dealt with in a global manner through the continuum.

Individual strand dynamics is computed to capture geometry and stiffness of each hair strand (see Section 2.2.1). Interaction dynamics, including hair-hair, hair-body, and hair-air interactions, are modeled using fluid dynamics. Individual hair strands are kinematically linked to fluid particles in their vicinity. Modeling a continuum requires defining the physical properties of the hair medium, such as density and pressure, at each point in the specified region. In this model, the density of the hair medium is defined as the mass of hair per unit occupied volume and the pressure and viscosity represent all of the forces due to interactions of hair strands. As hair strands are compressed together, the pressure increases and results in the strands being moved apart. Figure 2.7 shows the Eulerian and Langrangian viewpoints of strands in a fluid flow model.

Using this setup, it is possible to model hair-body interactions by creating boundary fluid particles around solid objects showing hair blowing in the wind. A fluid particle, or Smooth Particle Hydrodynamics (SPH), then exerts a force on the neighboring fluid particles based on its normal direction. The viscous pressure of the fluid, which is dependent on the hair density, accounts for the frictional interactions between hair strands. Hair-air interactions may also be captured using this method in order to add volume to the hair or to model the effects of wind on the hair. The hair medium is viewed as a mixture of hair material and air, each having its own fluid dynamics, which are linked by adding extra drag forces.
Utilizing fluid dynamics to model hair captures the complex interactions of hair strands. However, since this method makes the assumption of a continuum for hair, it does not capture dynamic clustering effects that can be observed in long, thick real hair. Moreover, computations required for this method are quite expensive; using parallelization, it took several minutes per frame to simulate a hair model composed of 10,000 individual hair strands.

**Loosely Connected Particles**

Bando *et al.* [BCN03] have modeled hair using a set of SPH particles that interact in an adaptive way. Each particle represents a certain amount of hair material which has a local orientation (the orientation of a particle being the mean orientation of every hair strand covered by the particle), refer to Figure 2.8.

Initially, connected chains are settled between neighboring particles being aligned with local hair orientation; two neighboring particles having close directions and being aligned with this direction are linked (all the more strongly as their orientations are close). This initial configuration is kept during the motion because it represents spatial consistency of interactions between particles. During motion, each particle can interact with other particles belonging to its current neighborhood. The method proposes to handle these interactions by settling breakable links between close particles; as soon as the two particles are not close enough, these links vanish. This method thus facilitates transversal separation and grouping.
Figure 2.8: (top) Particles defining hair, line segments indicate direction (bottom) Animation of hair with head shaking [BCN03].

while maintaining a constant length for hair. At each time step, searching the neighborhood of each particle is done efficiently by using a grid of voxels.

Interpolation between Guide Hair Strands

Chang et al. [CJY02] created a system to capture the complex interactions that occur among hair strands. In this work, a sparse hair model of guide strands, which were first introduced in [DTKT93, KAT93], is simulated. A dense hair model is created by interpolating the position of the remaining strands from the sparse set of guide strands. Using multiple guide hair strands for the interpolation of a strand alleviates local clustering of strands.

The sparse set of guide strands is also used to detect and handle mutual hair interactions. Since detecting collisions only among the guide strands is inefficient, an auxiliary triangle strip is built between two guide strands by connecting corresponding vertices of the guide strands. A collision among hair strands is detected by checking for intersection between two
hair segments and between a hair vertex and a triangular face. Damped spring forces are then used to push a pair of elements away from each other when a collision occurs.

Breakable connections, referred to as static links, are used to enable hairstyle recovery. The links are placed between nearby guide strands to capture the elastic lateral motion of hair and are initially placed based on the adjacency configuration of the described hairstyle. Under strong forces, these links, modeled as spring forces, are permanently broken, allowing the volume of hair to break into smaller pieces. A static link is broken once the distance between the corresponding guide strands surpasses a designated threshold and remains broken for the rest of the simulation.

The use of guide strands can lead to missed collisions when the interpolated strands collide with an object with which the guide strands do not.

**Free Form Deformation**

To achieve hair simulation of complex hairstyles in real-time, Volino et al. [VMT04] proposed to use a global volumetric free form deformation (FFD) scheme instead of considering an accurate mechanical model related to the structure of individual hair strands. A mechanical model is defined for a lattice surrounding the head. The lattice is then deformed as a particle system and hair strands follow the deformation by interpolation. Collisions between the hair and the body are handled by approximating the body as a set of metaballs.

This method is well-suited for animating various complex hairstyles, when the head motion has a low magnitude. For high deformations, hair discontinuities observed in real hair would not be reproduced, as only continuous deformations of hair are considered through the lattice deformation.

**Adaptive Techniques within a Continuum**

Modeling hair using a continuous medium can lead to missing clustering effects of hair strands that occur in nature. Though hair strands tend to move in accordance with nearby strands, neighboring strands can either move too far apart to continue to act alike or they can move closer to alternate strands and behave more closely with a new group of neighboring
strands.

Chang et al. [CJY02] propose an adaptive guide strand generation scheme that adds new guide strands to overly interpolated regions. A new guide strand is created when the distance between two neighboring guide strands becomes larger than a given threshold. This process helps to distribute accuracy on the fly where needed the most.

Bando et al.'s [BCN03] method of adaptively connecting hair particles can also achieve dynamic grouping and separation of hairs. Links between particles are either created or broken depending on the neighborhood of the particles at a given time. As a result, hair strands can change groupings depending on adjacency information.

2.2.3 Hair as Disjoint Groups

In order to reduce the complexity of hair, an alternative approach consists of grouping nearby hair strands and simulating these disjoint groups as independent interacting entities. This representation of hair was especially used to save computation time in comparison with the simulation of individual strands, and even reach interactive frame rates. It also captures realistic features of hair as it accounts for local discontinuities observed inside long hair during fast motion; these local discontinuities cannot be captured using the continuum paradigm.

Real-time Simulation of Hair Strips

As discussed in Section 2.1.2, the complexity of hair simulation has been simplified by modeling groups of strands using a thin flat patch, referred to as a strip (see Figure 2.2) [KH00, KN00, KH01, GZ02, LH03, SYAM05, TG05]. A simple dynamics model for simulating strips is presented in [KH01] that is adapted from the projective angular dynamics method introduced by Anjyo et al. [AUK92] (see Section 2.2.1); dynamics is applied to the control point mesh of the NURBS surface.

Using strips to model hair results in significantly faster simulation because fewer control points are required to model a strip in comparison to modeling individual strands. In [KH01] collision avoidance between hair strips and external objects (such as the head or the body) is achieved by using ellipsoids to approximate the boundaries of these objects. When a control
point of the strip is inside the ellipsoid a reaction constraint method is used to move it back to the boundary. Furthermore, collisions between hair strips are avoided by introducing springs within the strips and between neighboring strips. The springs are used to prevent neighboring strips from moving too far apart or too close together. Moreover, springs are also used to prevent a strip from overstretching or over-compressing. The result is that the hairstyle remains relatively consistent throughout the simulation.

By using a single strip to represent tens or hundreds of hair strands, hair simulation, including hair-hair collision avoidance, can be achieved in real-time. This process, however, is limited in the types of hairstyles and hair motions it can represent. The flat shape of the strips is most suited to simulating simple, straight hair. Curly, voluminous hair or complex hair motions that involve intricate hair interactions are not accurately captured with these methods.

Simulation of Wisps

One of the first methods taking advantage of grouping hair was presented by Watanabe and Suenaga in [WS92]. They animate a set of trigonal prism-based wisps. During motion, the shape of a wisp is approximated by parabolic trajectories of fictive particles initially located near the root of each wisp. At each time step, the trajectories of the particles are estimated using initial velocities and accelerations (such as gravitational acceleration). This method amounts to simulating only approximate kinematics without considering inertia of the system, which appears to be limited to slow hair motion. Moreover, interactions between different wisps are not taken into account.

A similar process of grouping neighboring strands together into wisps was used by [KAT93, DTKT93]. In these works, a wisp of strands is modeled by simulating the motion of a single typical strand and then generating other strands by adding random displacements to the origin of the typical strand. The number of overall strands that need to be simulated is reduced significantly. Again, in this work, interactions among strands, or between wisps, is not considered.
Processing Interactions between Deformable Volumetric Wisps

To account for complex interactions being observed in real hair during fast motion, Plante et al. [PCP01, PCP02] have introduced a layered wisp model. In this model, hair is represented as a fixed set of volumetric wisps. Each wisp is structured into three hierarchical layers: a skeleton curve that defines its large-scale motion and deformation; a deformable volumetric envelope that coats the skeleton and accounts for the deformation of the wisp sections around it; and a given number of hair strands that are distributed inside the wisp envelope and that are only used at the rendering stage of the process.

As the skeleton approximates the average curve of a wisp, it is likely to stretch or compress a bit while the wisp is not completely straight. The mass-spring simulation is thus better-suited for simulating wisps than the approaches striving to simulate inextensible individual hair strands (see Section 2.2.1).

Assuming that the local discontinuities inside hair are caused by collisions between wisps of different orientations, the method provides a model of anisotropic interactions between wisps. Wisps of similar orientations are allowed to penetrate each other, and are submitted to viscous friction, whereas wisps of different orientations actually collide in a very dissipative way.

This method has led to very realistic results for fast motions, capturing the discontinuities that can be observed in long, thick hair (see Figure 2.9). Nevertheless, very expensive computations were required for the examples shown, which was mainly due to the high cost for detecting collisions between the deformable wisps. Particularly for a very smooth hairstyle, this method causes needless computations as a very large number of interpenetrating wisps is then required for the simulation, which results in the process of many vain micro-collisions when the whole hair is at rest.

Adaptive Clustering

Bertails et al. [BKCN03] introduced an adaptive animation control structure, called the Adaptive Wisp Tree (AWT), that enables the dynamic splitting and merging of hair clusters.
Figure 2.9: The layered wisp model captures both continuities and discontinuities observed in long hair motion [PCP01].

The AWT depends on a complete hierarchical structure for the hair. The AWT represents at each time step the wisps segments of the hierarchy that are actually simulated (called active segments). The splitting process locally refines the hair structure when a given wisp segment is not sufficient for capturing the local motion and deformation. The merging process simplifies the AWT when the motion becomes coherent again. Splitting of wisp segments always begins at the tips of the hair strands. Splitting and merging is decided by the velocity of the wisp segments.

2.3 Hair Rendering

Realistic rendering of human hair requires the handling of both local and global hair properties. Local hair properties define the way individual hair fibers are illuminated individually. Section 2.3.1 describes the scattering properties of hair and reviews the different models that have been proposed to account for those 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 2.3.2. Global hair properties also include the way hair fibers cast shadows on each other; this issue
of self-shadowing, handled in Section 2.3.3, plays a crucial role in volumetric hair appearance.

### 2.3.1 Light Scattering in Hair

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

### The Nature of Hair

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 2.10), and the medulla, a structure of unknown function that sometimes appears near the axis of the fiber. The cross sectional shape varies from circular to elliptical to irregular [Rob94].

![Figure 2.10: An electron micrograph of a hair fiber that shows the structure of the outer cuticle surface, which is composed of thin overlapping scales [Rob94]. In this image, the fiber is oriented with the root at the top and the tip at the bottom.](image)

A good deal 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.

**Notation and Radiometry of Fiber Reflection**

The notation for scattering geometry is summarized in Figure 2.11. We refer to the plane perpendicular to the fiber as the *normal plane*. The direction of illumination is $\omega_i$, and the direction in which scattered light is being computed or measured is $\omega_r$; both direction vectors point away from the center. We express $\omega_i$ and $\omega_r$ in spherical coordinates. The inclinations with respect to the normal plane are denoted $\theta_i$ and $\theta_r$ (measured so that 0 degree is perpendicular to the hair). The azimuths around the hair are denoted $\phi_i$ and $\phi_r$, and the relative azimuth $\phi_r - \phi_i$, which is sufficient for circular fibers, is denoted $\Delta\phi$.

![Figure 2.11: Notation for scattering geometry](MJC+03)

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 projected solid angle of the scattered flux that results from a narrow incident beam from the direction $\omega_i$. It is defined as the ratio of surface radiance (intensity per unit projected area) exiting the surface in direction $\omega_r$ to surface irradiance (flux per unit area) falling on the surface from a differential solid angle in the direction $\omega_i$:

$$f_r(\omega_i, \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)}.$$
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+03]. 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 \( \omega_r \) to curve irradiance (flux per unit length) falling on the curve from a differential solid angle in the direction \( \omega_i \).” 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+03] 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, the two will have the same scattering function but the wider fiber will produce a brighter curve in a rendered image, because the 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) and the factor of \( \cos \theta_i \) is often included in the model, as was common in early presentations of surface shading models, rather than moved into the definitions.
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], 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 all lie in the normal plane, a plane perpendicular to the fiber. 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 corollary of Snell's Law that is often used to describe refractions through crystals with 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.

Measurements of Hair Scattering

In the 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, providing 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+03] 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 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, confirming quite clearly 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 angle of incidence and that show the full scattered distribution for a particular angle of incidence.

Models for Hair Scattering

The earliest and most widely used model for hair scattering is Kajiya and Kay’s model, 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 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 $\theta$ or $\phi$.

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.\footnote{Banks does not discuss why uniform curve radiance is the appropriate sense in which the scattered light should be uniform.} 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 $f_{\text{dir}}$ that can be expressed in the current notation as:

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

Setting $a > 0$ serves to bias the model toward backward scattering, while setting $a < 0$ biases...
the model towards forward scattering.\textsuperscript{2}

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

\[
f_{\text{dir}} = a g(\Delta\phi/2) + g(k(\Delta\phi - \pi))
\]

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 \textit{et al.} [MJC+03] 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. Figure 2.12 illustrates the Kajiya and Kay model, Marschner \textit{et al.}'s recent scattering method, and real hair.

\textsuperscript{2}In Goldman's original notation \(a = (\rho_{\text{reflect}} - \rho_{\text{transmit}})/(\rho_{\text{reflect}} + \rho_{\text{transmit}})\). A factor of \(\frac{1}{2}(\rho_{\text{reflect}} + \rho_{\text{transmit}})\) can be absorbed into the diffuse and specular coefficients.
2.3.2 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.

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 [NvdP98] 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.

**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 [YXWY00] 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, the method does not allow changes in the density functions, making hairs appear as if they always stay together.

### 2.3.3 Hair Self-Shadowing

Hair fibers cast shadows onto each other, as well as receiving and casting shadows from/to other objects in the scene. Self-shadowing creates crucial visual patterns that distinguish one hairstyle from another. 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+03]. 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.
Ray-casting through a Volumetric Representation

With implicit hair representations, one can directly ray trace volume density [YXWY00], or use two-pass shadowing schemes for volume density [KK89] where the first pass fills volume density with shadow information and the second pass renders the volume density.

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:

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

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

![Image](image.png)

Figure 2.13: Opacity Shadow Maps. (left) Hair volume is uniformly sliced perpendicular to the light direction into a set of planar maps storing alpha values. (right) The resulting shadowed hair [KN01].

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. By approximating the transmittance function with discrete planar maps, opacity maps can be efficiently generated with graphics hardware (see Section 2.3.4). Linear interpolation from such maps facilitates fast approximation to hair self-shadows.
2.3.4 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.

Approximating Hair Geometry

Section 2.3.1 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].

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. Moreover, it can easily be parallelized to increase per-
formance.

**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 2.3.3), 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 2.3.4), 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.

### 2.4 Model Simplification

Model simplification algorithms, such as automatic generation of geometric level-of-detail (LOD) representations and multi-resolution modeling [SZ98] techniques, have been proposed to accelerate the rendering of complex geometric models. A recent survey on polygonal model simplification is presented in [Lue01]. A generic framework for selecting and
switching between different geometric levels-of-detail (LODs) to attain a nearly constant frame rate for interactive architectural walkthroughs was introduced in [FS93].

The use of levels-of-detail has been extended to motion modeling and dynamic simulation as well. Simulation levels-of-detail (SLOD) are used to simplify or approximate the dynamics in a scene, similar to the way that geometric LODs are used to simplify a complex model.

Simulation levels of detail can be generated from either pre-recorded motion sequences, procedural approaches, kinematics, or based on dynamics computation [BC89, GM85, KB93, Per95]. Some of the earlier human motion models in computer animation exploited this concept implicitly by using procedurally generated motion, simplified dynamics and control algorithms, off-line motion mapping, or motion play-back [BC89, GM85, KB93, Per95].

Carlson and Hodgins explored techniques for reducing the computational cost of simulating groups of legged creatures when they are less important to the viewer or to the action in the virtual world [CH97]. Though the generation of SLODs, switching and selection are designed by hand, this work shows the potential of automatic simplification of general dynamical systems.

In [PC01], levels-of-detail, including 3D geometry, volumetric textures and 2D textures, are used to animate and render prairies in real-time. SLODs have also been proposed for the automatic dynamics simplification of particle systems [OFL01]. This was accomplished by generating a physically-based subdivision of the particle system to create a SLOD-tree, which was updated on the fly.

Other types of simulation acceleration techniques, such as view-dependent dynamics culling [CF97] and Neuro-Animator [GTH98], have also been investigated to reduce the total computational costs for simulating a large, complex dynamical system.
Chapter 3

Level-of-Detail Framework for Modeling Hair

In this chapter, I present the basic framework for level-of-detail hair modeling, which involves choosing the geometric representation of hair as well as simulation and rendering schemes. The geometric representation of hair dictates the simulation, rendering, and styling methods for the hair, thus, becoming a crucial first step to hair modeling. As discussed in Chapter 2, 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. Often hair is modeled as groups of strands, or wisps, where multiple rendered strands are animated and styled as larger groups. These disjoint hair groups are also modeled as strips of hair through two-dimensional surfaces. Hair has also been perceived as one large volume; animation and styling is then controlled through a continuous medium while either one-dimensional strands or surfaces can be used to render the volume of hair.

Traditionally, these separate hair modeling representations either offer high simulation quality or high 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 the simulation speed and the simulation quality. In this chapter, I will describe the three representations I use to model hair. These representations, which I also refer to as the discrete levels-of-detail for modeling hair include individual strands, clusters, and strips, see
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.

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 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].

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 sys-
tem without any extra expensive computations other than performing the LOD switching tests.

Furthermore, the subdivision framework has been adapted for each LOD because it can model different hair shapes as effectively as NURBS, quickly perform adaptive dynamic tessellation, and can be used easily with new graphics hardware for interactive rendering. The subdivision scheme used for each LOD representation will also be described in this chapter.

Hair simulation is controlled through the use of the base skeleton. I will present a method that uses implicit integration to control the bending of the skeleton, which manages the motion of each LOD representation. Moreover, I introduce a collision detection method that efficiently and correctly handles hair-object and hair-hair interactions. The remaining part of the LOD framework involves hair rendering. I have adapted previous methods that capture intricate light scattering and self-shadowing characteristics and have tailored them for efficient rendering using each LOD representation; these techniques will also be explained in this chapter.

Finally, I will 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. I 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, I will discuss how these criteria are used together to choose the final representation for the hair, show results and discuss the performance of these methods.

3.1 The Base Skeleton

The base skeleton controls both the shape and the motion of each hair representation at any given point during simulation. This section explains the use of the base skeleton in dictating the shape of each LOD, while Section 3.4 will explain the simulation of the skeleton in detail.

Based on the idea for modeling each individual hair strand [AUK92, KAT93], a structure
Figure 3.2: (a) The base skeleton model (b) The parameters that define the shape of hair.

has been employed for the base skeleton that forms the core of the proposed set of LOD representations. The 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 the simulation. Figure 3.2(a) shows the basic setup of the skeleton. The \( n \) nodes \((P_0, P_1, \ldots, P_{n-1})\) and \( n - 1 \) segments \((s_1, s_2, \ldots, s_{n-1})\) define the skeleton.

**Shape Specification**

The base skeleton directs the placement of the control vertices used in subdivision for each hair representation. Various shapes or styles of hair are then specified by stipulating the rest angles \( \theta_{i0} \) and \( \phi_{i0} \) of each node \( i \) of the skeleton. Figure 3.2(b) shows definitions for
θ₁₀ and φ₁₀, resting angles for node Pₙ. The line segment sₙ₋₁, defined by nodes Pₙ₋₁ and Pₙ₋₂, defines the y-axis for the local reference frame of node Pₙ. Polar angles θ₁₀ and φ₁₀ then define the position of Pₙ in relation to line segment sₙ₋₁.

Control point P₀ is defined as the root node; it is always attached to a designated location on the scalp and is never moved from that location during subsequent motions. Control point P₁ is always offset from the root node by a small distance in the normal direction from the root node on the scalp. The positioning of node P₁ slightly normal to the head gives a little extra volume to the hair. As a result skeletal segment s₁ created between control points P₀ and P₁ always remains normal to the scalp and is not subject to dynamic simulation or collision detection. Figure 3.3 shows the base skeleton attached to the scalp at the root node.

The base skeleton shape created from the rest angles and resulting collisions with the head and among hairs are referred to as the hair’s resting style, meaning the shape of the hair when there are no external forces, such as wind, influencing the hair’s form. Throughout the dynamic simulation, the positioning of a base skeleton is determined from its rest angles, collisions, and the forces being applied to the hair, such as gravity, wind, etc. Section 3.4 will explain dynamic simulation and collision detection along with their effects on hair positioning.

A resting style of straight hair can be created by assigning θ₁₀ to 0 and φ₁₀ to 0 for each
node $P_i$ of the skeleton. In addition, a wavy hairstyle can be created by zig-zagging the position of the nodes down the length of the skeleton. A zigzag or wavy skeleton is created by assigning each $\theta_{i0}$ to a certain angle between 0 and 90 degrees, depending on the severity of the wave, and then the values of $\phi_{i0}$ alternate by 180 degrees. Ringlet or spiral curls can also be created using the skeleton by specifying an angle value between 0 and 90 degrees for $\theta_{i0}$ and then, to achieve the spiral effect, each $\phi_{i0}$ value increments by 90 degrees down the length of the skeleton. Three basic hairstyles and a horse’s mane and tail created with this system are shown in Figure 3.4

These two processes for stipulating waves or curls can be altered to create varying styles. The segment sizes, $s_i$, and the values for $\theta_{i0}$ and $\phi_{i0}$ can be changed, or chosen randomly, to create non-uniform curls and waves according to the desired outcome.

### 3.2 Geometric Representations

Using the base skeleton, the three discrete hair representations are then 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.

#### 3.2.1 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.
Figure 3.4: Different hairstyles generated using the LOD representations and the base skeleton. From left to right, top to bottom: (1) Short, straight, blonde hair (2) Short, wavy, brown hair (3) Long, curly, blonde hair (4) Horse mane and tail.

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 3.6.

3.2.2 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.

### 3.2.3 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 still 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.
Figure 3.5: Cross-section of a strand group and placement of a strand within the group. The skeleton node is at the center of the circular cross-section. The strand placement (shown as the white dot) is determined by its angular placement and percentage of the radius distance value.

When grouping hair strands to follow the same base skeleton, the number of strands in the group can vary depending on the simulation. Chapter 4 explains the algorithm I have developed to automatically merge and split groups of strands on-the-fly. For now, we will allow the definition of a strand group to consist of a base skeleton, a radius defining the boundaries of the group within which hair strands are placed (this value can vary down the length of the skeleton), and finally the number and placement of strands that will follow the base skeleton.

Once the skeleton and radius have been defined for a strand group, individual strands are randomly placed on the scalp within the boundaries of the circular radius around the skeleton root node, Figure 3.5 illustrates this process. Each strand possesses its own values to maintain a consistent placement within the strand group down the length of the skeleton. A strand is given an angle, from 0 to 360 degrees, to place it around the skeleton and a percentage value to define its distance from the skeleton node. Finally, the length of each strand in a group should vary slightly for added realism since in real-life strands are rarely all the same length. The number of control points remains the same, the length will just vary so that each strand ends between skeleton control points \( P_{n-2} \) and \( P_{n-1} \), where \( n \) is the number of control points...
in the skeleton.

3.3 Subdivision Representations

Subdivision curves and surfaces have been chosen as the underlying geometric representation for all LODs in the hair modeling framework because of their scalability and uniformity of representation [SZ98]. The subdivision process creates smooth curves and surfaces through successively refining a curve or mesh of control points. Defining the levels of successive refinement can control the smoothness of the resulting surface or curve. This is used to generate adaptive, continuous LODs for rendering. A detailed discussion on the subdivision framework and techniques can be found in [SZ98].

Subdivision in 1D is used to create the curves that represent hairs as individual strands, as discussed in Section 3.2.3. The 4pt scheme in 1D is an efficient method for creating a smooth curve. For surface representations, a similar process is employed. A strip is created from two parameterized line segments, with the skeleton in the center, that are connected to create a triangular mesh, see Figure 3.1(a). Due to the uniformity in motion created from the skeleton, the two line segments will always move in a coherent manner, meaning the triangular mesh will never stretch or compress laterally. As a result, a complex surface subdivision technique is not necessary to capture the detail of the moving strip. It is sufficient to perform the 4pt scheme on each of the parameterized line segments forming the strip.

Similarly, the same process is exercised for cluster subdivision. The 4pt scheme can be applied efficiently to each of the $m$ lines of control vertices created to make the cluster in Section 3.2.2. The control meshes and final rendered versions for each of the three representations are shown in Figure 3.1.

3.4 Hair Simulation

Hair simulation is composed of two essential parts: dynamic hair motion and collision detection. The motion of the hair is controlled through the simulation scheme chosen. The great complexity of hair simulation is caused by the high number of deformable hair strands
that typically comprise the human head. These hair strands are in constant contact with other strands as well as the human avatar. Computing all possible contacts of these thin, deforming strands is far too overwhelming for most systems to handle. As a result, many hair modeling schemes compromise accuracy and visual quality in order to attain a desirable performance, as explained in Chapter 2. Achieving realistic hair motion has been the most challenging aspect of hair modeling, especially for systems that have high performance demands.

My work in hair simulation has focused on creating a system that can realistically model many complex features of hair motion, such as dynamic clustering effects, while accelerating the computations of features that have traditionally been the bottleneck of hair modeling, e.g. computing hair-hair interactions. In this section, I will explain the efficient simulation techniques that I have developed for level-of-detail hair modeling. They include a dynamics control structure developed for the base skeleton, an implicit integration scheme designed specifically for hair motion, and a collision detection and response framework built for the structure of hair. Figure 3.6 shows an illustration of wind blowing through long hair.

3.4.1 Single-Skeleton Dynamics

As explained in Chapter 2, traditional hair modeling techniques have often employed a single-skeleton model to control the shape and motion of multiple hair strands or of a single hair strip [AUK92, KAT93, PCP01, KH01]. The methodology was developed from the observation that hairs in close proximity tend to move similar to each other. Using a single skeleton to control a group of hair captures realistic hair grouping and is far more efficient.
than simulating each rendered strand individually.

The dynamics of a single-skeleton system can be used to control each of the level-of-detail representations. Automatically the LOD structure creates a more efficient dynamics system over traditional strand or wisp modeling schemes. Established wisp structures [WS92, KAT93, DTKT93, PCP01, CCK05] use a static number of skeletons to control the groups of hair, no matter the complexity of the simulation. Since a strip can represent more hair than a single cluster and a single cluster can model more hair than a single strand, when the system is at a lower LOD, there are fewer skeletons to simulate, increasing the simulation performance. This refinement is only one way the LOD techniques accelerate simulation, however more detail on LOD simulation refinement and performance comparisons will be given in Section 3.7. For now, this section will explain the dynamics of a single-skeleton system that is applicable to each LOD representation including the implicit integration that allows for greater stability when modeling stiff springs.

The single-skeleton system dynamics start from the basic dynamics model for simulating hair that was first proposed by [AUK92, KAT93]. Here, projective angular dynamics are used to control the angles between successive control points on the hair strand, or skeleton model. The skeleton is modeled as an open chain of rigid line segments connecting node points. Spring forces are used to control the angles between the nodes, while the distance between each node is fixed. I have extended this method by using an implicit integration technique to control the bending motion of the skeleton in order to achieve greater stability while allowing the system to take larger time steps throughout the simulation. This approach is similar to cloth simulations that use implicit integration for greater stability [BW98].

In this approach, each control point of a hair skeleton is governed by the set of ordinary differential equations:

\[
I_i \frac{d^2 \theta_i}{dt^2} + \gamma_i \frac{d\theta_i}{dt} = M_{\theta i} \quad (3.1)
\]

\[
I_i \frac{d^2 \phi_i}{dt^2} + \gamma_i \frac{d\phi_i}{dt} = M_{\phi i} \quad (3.2)
\]

where \( I_i \) is the moment of inertia for the \( i \)th control point of the skeleton, \( \gamma_i \) is the damping
coefficient, and \( M_{\theta i} \) and \( M_{\phi i} \) are the \( \theta \) and \( \phi \) torque components, respectively. \( M_{\theta} \) and \( M_{\phi} \) are computed from the spring forces controlling the style of the hair and external forces such as wind.

![Diagram](image)

**Figure 3.7:** Bending motion of a hair skeleton (a) Current positioning of \( \theta_i \) (b) Goal, or rest, position, \( \theta_{i0} \).

Figure 3.7 illustrates the use of spring forces to control the angles of the skeleton. The goal of the spring forces is to return the hair to its original resting style. Thus, if curly hair begins to move due to wind, it should try to retain and then return to the same curled shape when the wind force is removed. The torque equations due to spring forces are calculated by:

\[
M_{\theta i} = -k_{\theta}(\theta_i - \theta_{i0}) \tag{3.3}
\]

\[
M_{\phi i} = -k_{\phi}(\phi_i - \phi_{i0}), \tag{3.4}
\]

where \( k_{\theta} \) and \( k_{\phi} \) are the spring constants for \( \theta \) and \( \phi \), respectively. Furthermore, \( \theta_{i0} \) and \( \phi_{i0} \) are the specified rest angles and \( \theta_i \) and \( \phi_i \) are the current angle values.

**Implicit Integration**

Although explicit methods such as Euler or fourth-order Runge-Kutta 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 required to avoid instability. The implicit integration scheme not only offers greater stability, but also provides a generality to modeling more diverse hairstyles over the aforementioned explicit techniques. This implicit derivation for hair modeling was first presented in Ward and Lin [WL03].

I will first show how the implicit scheme is derived for the $\theta$-component. Because the bending motion is measured in polar coordinates, the equations will display angular positions, $\theta$ and $\phi$, angular velocities, $\omega_\theta$ and $\omega_\phi$, and angular accelerations, $\alpha_\theta$ and $\alpha_\phi$.

Rewriting Equation 3.3 as a second-order differential equation returns:

$$\ddot{\theta}(t) = f(\theta(t), \dot{\theta}(t)) = -k_\theta(\theta_i - \theta_0). \tag{3.5}$$

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:

$$\begin{pmatrix} \omega_\theta \\ \alpha_\theta \end{pmatrix} = \frac{d}{dt} \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = \frac{d}{dt} \begin{pmatrix} \theta \\ \omega_\theta \end{pmatrix} = \begin{pmatrix} \omega_\theta \\ f(\theta, \omega_\theta) \end{pmatrix}. \tag{3.6}$$

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

$$\begin{pmatrix} \Delta \theta \\ \Delta \omega_\theta \end{pmatrix} = h \begin{pmatrix} \omega_{\theta 0} \\ -k_\theta(\theta - \theta_0) \end{pmatrix}. \tag{3.7}$$

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:

$$\begin{pmatrix} \Delta \theta \\ \Delta \omega_\theta \end{pmatrix} = h \begin{pmatrix} \omega_{\theta 0} + \Delta \omega_\theta \\ f(\theta_0 + \Delta \theta, \omega_{\theta 0} + \Delta \omega_\theta) \end{pmatrix}. \tag{3.8}$$

A Taylor series expansion is applied to $f$ to obtain the first-order approximation:
\[ f(\theta_0 + \Delta \theta, \omega_{\theta 0} + \Delta \omega_{\theta}) \approx f_0 + \frac{\partial f}{\partial \theta} \Delta \theta + \frac{\partial f}{\partial \omega_{\theta}} \Delta \omega_{\theta} \]
\[ \approx -k_\theta (\theta - \theta_0) - k_\theta \Delta \theta + 0(\Delta \omega_{\theta}) \approx -k_\theta (\theta - \theta_0) - k_\theta \Delta \theta \]  
(3.9)

Substituting the approximation of \( f \) back into the differential equation of Equation 3.8 yields:

\[
\begin{pmatrix}
\Delta \theta \\
\Delta \omega_{\theta}
\end{pmatrix} = h
\begin{pmatrix}
\omega_{\theta 0} + \Delta \omega_{\theta} \\
-k_\theta (\theta - \theta_0) - k_\theta \Delta \theta
\end{pmatrix}.
\]  
(3.10)

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

\[ \Delta \omega_{\theta} = h(-k_\theta (\theta - \theta_0) - k_\theta h(\omega_{\theta 0} + \Delta \omega_{\theta})) \]

Rearranging this equation gives:

\[ (1 + k_\theta h^2) \Delta \omega_{\theta} = -h k_\theta (\theta - \theta_0) - k_\theta h^2 \omega_{\theta 0} \]
\[ \Delta \omega_{\theta} = \frac{-h k_\theta (\theta - \theta_0) - h^2 k_\theta \omega_{\theta 0}}{1 + h^2 k_\theta}. \]  
(3.11)

The change in angular velocity for the \( \theta \)-component of a skeleton node point, \( \Delta \omega_{\theta} \), is thus given in Equation 3.11, where \( h \) is the time step, and \( \omega_{\theta 0} = \omega_{\theta}(t_0) \) is the angular velocity at time \( t_0 \). Once \( \Delta \omega_{\theta} \) has been calculated, the change in angular position, \( \Delta \theta \), is calculated from \( \Delta \theta = h(\omega_{\theta 0} + \Delta \omega_{\theta}) \). The same process is applied to the \( \phi \)-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.
3.4.2 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. This complexity arises from the sheer number of deformable strands that are in constant contact with each other and the body of the avatar. I have separated the collision algorithm into two parts: *hair-object* collision detection, which occurs when hair interacts with an impenetrable object like the head, and *hair-hair* collision detection, involving hair mutual interactions. Hair-hair interaction differs from hair-object interaction in that some penetration of the hairs may be permitted depending on a number of factors that will be described later in this section.

Collision detection is a vital part of hair simulation. Obviously hair-object collision handling is necessary to prevent the hair from erroneously passing through the head and body - a feature forbidden to any hair modeling algorithm whether it be high quality feature films or high performance virtual environments. Handling hair-hair interactions controls many key elements for depicting hair. The volume of the hair is defined by hairs interacting with and resting on top of each other. Also, many hairstyles are defined by how the strands lay in contact with each other, such as braids or twists, which require a precise method for computing hair collisions. Finally, it can be visually distracting to observe strands of hair awkwardly passing through each other in a way that is not possible in the real world. Though it can be computationally expensive, hair-object and hair-hair collision detection and response are crucial to a realistic hair modeling system.

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 efficient overall collision
detection method.

In the rest of this section, I will describe the novel selection of appropriate bounding volumes for each LOD representation. Then, I 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.

I have chosen to utilize the family of “swept sphere volumes” (SSVs) \cite{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 3.8 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:

\[
B = C \oplus S = \{ c + r \mid c \in C, r \in S \}.
\]  

(3.12)

![Figure 3.8: 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.](image)
To detect an intersection between a pair of arbitrary SSVs a distance test is performed between their corresponding core skeletons and then the appropriate offsets, i.e. the radius of each SSV, are subtracted.

**Swept Sphere Volumes for Hair**

I 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 an 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, as illustrated in Figure 3.9, which shows a section of a strip compared with its cluster counterparts. Thus, when a transition from a strip into clusters occurs, the volume of hair being represented remains consistent.

For the cluster representation, an LSS is created around the \( 2m \) control points that define a segment (\( m \) control points, as defined in Section 3.2.2, 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.
Figure 3.9: Strip is rendered as a surface but acts like a volume. (top) Cross-sections of three clusters line up with cross-section of strip (shown as brown line); Radius of the clusters used to determine offset of RSS for strip (bottom) Comparison of three clusters and strip with offset, bold black lines indicate volume created for the strip.

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 collision 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 3.10: Overlap of two line swept spheres (LSSs). (left) Compute distance \( d \) between core lines (right) Subtract offsets to determine overlap value.](image)

For each pair of SSVs that falls into the same grid cell the distance between their corresponding core skeletons, \( s_1 \) and \( s_2 \), are determined. This distance, \( d \), is subtracted from the sum of the radii of the two SSVs, \( r_1 \) and \( r_2 \), to determine if there is an intersection. Let

\[
\text{overlap} = d - (r_1 + r_2)
\]  

(3.13)

If \( \text{overlap} \) is positive then the sections of hair do not overlap and no response is calculated.
Figure 3.10 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 [PCP01], the orientations of the two hair sections will determine how the collision response is handled. The cross product between the core skeletons, $s_1$ and $s_2$, is computed to determine the orientation of the skeletons in relation to each other. If $s_1$ and $s_2$ are near parallel, the velocity averaging will be enough to combat their collision, similar to [PCP01]. Whereas [PCP01] 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 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 $s_1$ to the closest point on $s_2$. Each section is moved by half the overlap value and in opposite directions along the vector from $s_1$ to $s_2$. Figure 3.11 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 I 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, $\mu_f$, which is dependent upon the surface of the object, where $0 < \mu_f < 1$. The resulting friction force, $F_f$, becomes:

$$F_f = -\mu_f(F - (F \cdot N)N)$$  \hspace{1cm} (3.14)

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

3.5 Hair Rendering

As detailed in Chapter 2, hair rendering is responsible for modeling hair's interaction with light. Anisotropic lighting is typically used to model the reflection of light from the hair's surface while self-shadowing details are important, yet complex and detailed to capture to give hair a realistic appearance. In this section, I will explain the rendering methods adapted for the LOD framework, which include capturing multiple modes of light scattering and self-shadowing effects, as well as the rendering methods used to represent each LOD and to transition between them.

3.5.1 Lighting Scattering

The anisotropic lighting effects are modeled as light scattering from a cylindrical surface as proposed by Kajiya and Kay [KK89] and augmented by recent observations made by Marschner et al. [MJC +03] that account for the multiple modes of scattering that occur inside and on the surface of the cylinders. Light scattering on human hair fibers was measured by [MJC +03], which showed that only a limited amount of colored reflection comes from diffuse scattering while there is a major contribution of uncolored specular reflection. For LOD hair rendering, two separate specular highlights are modeled to represent the multiple modes of
scattering inside and on the surface of the hair fibers. The primary highlight is highly specular and shifted slightly towards the hair root, whereas the secondary highlight has a wider falloff and is shifted towards the hair tip. Both specular terms are computed by shifting the hair tangent in Kajiya's original formulation [KK89] towards the hair root and towards the hair tip respectively, applying separate falloff exponents. Section 2.3 explains both Kajiya's model and Marschner's modifications in detail.

While the recent work of [MJC+03] provided an accurate model for light scattering on human hair, Heidrich and Seidel [HS98] showed that anisotropic lighting could be calculated efficiently on graphics hardware using an OpenGL texture matrix. In the LOD hair framework these two techniques have been combined to create an efficient light scattering model. Using the shifted tangents explained by [MJC+03] in the formulation proposed in [HS98], the new two-fold specular term of the hair shading thus becomes:

\[
I_{\text{specular}} = (k^1_s(V, R^1)^{n_1} + k^2_s(V, R^2)^{n_2}) \cdot I_i
\]

\[
\langle V, R^i \rangle = \sqrt{1 - \langle L, T^i \rangle^2} \sqrt{1 - \langle V, T^i \rangle^2} - \langle L, T^i \rangle \langle V, T^i \rangle
\]

where \(k^1_s, k^2_s\) are the specular reflection coefficients and \(n_1, n_2\) represent the surface roughness for each specular term, \(V\) is the unit vector towards the viewpoint, \(L\) is the unit vector towards the light, \(R^i\) is the reflection of \(L\) at normal \(N^i\), and \(T^i\) is the tangent used for each specular term. The tangents are shifted in the plane formed by the hair tangent and the light direction. Hair fiber roughness due to tilted surface scales is accounted for by multiplying the second highlight shift with a noise pattern. All operations are then performed in a fragment program for efficiency.

### 3.5.2 Hair Self-Shadowing

Realistic hair self-shadowing effects are hard to implement efficiently, due to the large amount of dynamic geometry and to the fineness of the hair strands. Regular shadow maps fail because the high frequency in the geometry forces the use of sampling rates beyond realistic values (both in terms of performance and memory usage). Two recently introduced features
of graphics hardware are used by the LOD framework to achieve fast self-shadows. The first, *multiple render targets*, allows 16 floating point values to be output in one rendering pass. The second, *floating point blending* makes high-precision blending of shadow values possible.

*Opacity shadow maps* (OSM) [KN01], as explained in Section 2.3, were a first attempt at efficient self-shadow generation for hair, but the algorithm is still not quite interactive. In this technique, self-shadows are generated by accumulating the opacity $\alpha$ of the strands hit by the light rays along the light direction in the hardware framebuffer. Unfortunately, because of the iterative nature of the algorithm, it requires an expensive multiple pass rendering algorithm with an expensive read-back to the CPU. The number of passes depends on the number of opacity maps, hence on the achieved image quality. Moreover, the accumulation of opacity through blending in the framebuffer was limited to 8-bit integers. This leads to artifacts due to limited precision. For 10 opacity shadow maps, this technique achieves a rate of less than one frame per second.

Our algorithm is based on recent GPU features [NVI05]; it generates 16 opacity shadow maps in only one pass with *multiple render targets*, plus an extra pass for the actual rendering, without depth ordering required. As in [KN01], the opacity maps are placed at uniformly sampled distances from the eye representing the light, orthogonal to the view direction of the light camera. Each of the four render targets holds four opacity maps, one in each 16-bit floating point component.

In the *opacity map generation pass*, a GPU fragment program compares the distance of the incoming hair fragment from the eye to the distance of each of the 16 maps from the eye, illustrated in Figure 3.12. If the hair fragment is closer to the eye than a particular opacity map, it means that it contributes to the opacity of the map in question, thus we give it a positive contribution. Otherwise, the contribution is zero. By enabling 16-bit floating point blending, we achieve high-precision accumulation of the opacity of all the fragments into the appropriate opacity maps.

In the *rendering pass*, a vertex program first computes weights according to the relative position of the vertex to each of the opacity maps. These weights are then used in the fragment program for reconstruction. The weights are computed as follows:
Figure 3.12: Shadow effects. Side-by-side comparison (left) without and (center) with hair shadows; (right) Light position and orientation is indicated with grey lines, OSM cutting planes are shown in red around the hair model.

\[ w_p^n = \frac{\max(0, 1 - |z_n - z_p|)}{\Delta}, \]  

(3.17)

where \( w_p^n \) is the weight of point \( p \) for the \( n^{th} \) opacity map, \( z_n \) and \( z_d \) are the distances of the \( n^{th} \) opacity map and point \( p \) to the eye. \( \Delta \) is the distance between the opacity maps.

Our method has an additional advantage over the technique proposed by [KN01]: instead of assigning the hair geometry to depth bins on the CPU, we perform the depth comparisons at the fragment level (in fragment programs on the GPU), therefore avoiding the expensive read-back to the CPU. In addition, we can use the opacity generation pass to generate a regular shadow map for the head, with no overhead incurred. The blocking of light by the head is then easily determined by regular shadow mapping techniques, and allows us to cast hair shadows on the head and the body. Simply sampling the OSM does not work, because the OSM contains no information whether the fragment is visible from the light or not, similar to projected textures. Instead, we combine the OSMs with visibility information obtained from the regular shadow map, and take the maximum of both values. The result is linearly interpolated hair shadows on the body and head. Figure 3.12 illustrates the importance of shadow effects in creating a hair model in comparison to no shadows on the hair.
Limitations

This approach can incorrectly make fully-lit fragments that lie between opacity maps appear darker because of the linear interpolation between the lit and non-lit opacity map plane. We solve this partially by virtually pushing the opacity map planes back, so that the fully-lit fragment samples lit planes with more preference. Another approach would be to use quadratic interpolation.

3.5.3 Level-of-Detail Hair Rendering

In the level-of-detail hair system, a mixed model of discrete and continuous geometric LODs is used to render the hair representations. Each LOD has a distinct geometric representation: a strand is rendered as a subdivision curve, a cluster is rendered as a ruled subdivision surface (i.e. a surface generated by sweeping a curve around an axis), and a strip is rendered as a subdivision surface. By varying the tessellation parameters as a function of the distance to the viewer, an almost continuous LOD is provided for each geometric hair representation.

As explained earlier, the anisotropic lighting is modeled with an efficient graphics hardware method using a texture matrix. Each hair strand is assigned a random shade within a range of specified hair colors. This value is then blended with the anisotropic lighting. The rendering of the subdivision surfaces that are used for clusters and strips are assigned two additional textures. The first texture contains the hair color information. It is created as a pre-process from the same color range that assigns color values to the strands. Thus, the hair colors of all three representations are the same. The next texture used on the surfaces contains alpha values that define the transparency of the surfaces, similar to the strips created by [KH00].

Alpha mapping is used to create the illusion that there are individual strands being rendered. Normally the clusters and strips have smooth, flat bottom lines and are completely solid in representation. Meanwhile, the individual strands do not look like a solid volume and they possess slightly varying lengths. We imitate these irregular lengths and create the illusion of strands via the alpha channel.
Self-shadowing is a vital factor to increase the volumetric cue of the hair and recent graphics hardware advances have helped to accelerate traditional shadow generation schemes. Using LOD representations also inherently saves time, because the number of points to represent the subdivision surface of a cluster or a strip is far less than that of the strands. Density values of each hair representation are chosen proportional to the number of strands each cluster and strip contains in order to guarantee the same brightness in different LODs. We adopt interactive lazy update so that the opacity maps do not need to be updated every frame. If hairs do not move very much, neither does the opacity map, so the opacity map update can be delayed. We also change the number of opacity maps interactively depending on the length of the hair. Long hairs require more maps to achieve a better look.

Aliasing is an innate problem of hair rendering, because each hair strand is too thin to occupy a single pixel. Fortunately it is not as significant a problem for high LOD representations like the clusters and strips, as it is for strands. Anti-aliasing for hair rendering requires pixel blending in back-to-front order [LTT91, KN02]. In our system, the simulation updates hair geometry every frame and the user can change the viewpoint frequently. Therefore, the hair geometry should be reordered from the camera point of view every frame with this method. Instead we rely on graphics hardware for anti-aliasing. The NVIDIA® GeForce™ family provides hardware implemented high resolution anti-aliasing through multi-sampling [NVI05]. We select 4-sample 9-tab multi-sample mode because various multi-sampling modes provided by the graphics hardware do not affect overall performance of the system.

In order to render the simulated hairs with motion blur, we adapt the technique presented by [HA90]. Using this method, the current image is rendered into the accumulation buffer so that it can be integrated with previous images. Using the accumulation buffer also reduces the aliasing of individual strands.

Switching between different LODs can introduce visual disturbances. To address this problem, we combine several methods. By using the alpha channel in the textures for clusters and strips we add the visual illusion of strands, making the LOD transition less noticeable. The density values for the clusters and strips are based on the number of strands they represent. This helps to keep the brightness of the shadows constant. We also blend the images of
previous and current LODs to make transitions smoother through the accumulation buffer.

3.6 Runtime Selection of Hair

Now that the main components of the level-of-detail framework for hair have 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 simulation 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 simulation.

The importance of a section of hair relates to the amount of 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 application and it is then simulated 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 a single strip, a number of \( c \) clusters, or a number of \( s \) strand groups, where \( c < s \). The section of hair at any point during the simulation will either be modeled as a strip, the clusters, or the strand groups, but never more than one representation at a time. Chapter 4 will explain the method I have developed for using varying resolutions of each representations, meaning \( c, s \), and the number of strips to represent a volume of hair will vary throughout the simulation and a section of hair can be modeled using a combination of the varying resolutions of LODs. The purpose of this section is to show how the use of the three discrete LODs alone can appreciably accelerate hair modeling.

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

- **Visibility** - Measures if the viewer can see a section of hair;

- **Viewing distance** - Measures how far a section of hair is from the viewer, which
correlates to how much screen space the hair covers;

- **Hair motion** - Measures the velocity of a section of hair to determine how much simulation detail 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.

### 3.6.1 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 \( s \) 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. Figure 3.13 illustrates the occlusion of hair.

A straightforward implementation to illustrate the contribution of this technique is to perform a simple occlusion test by fitting a sphere to the head so that it is slightly smaller than the head. If there are other objects in the scene, such as a body, a similar method is applied. Then the SSV bounding volumes of the hair are tested to see if they are visible from the camera or if the occluders occlude them. It is possible to use more sophisticated occlusion culling algorithms, such as [ZMHI97], or special features on new GPUs to perform these tests more efficiently; however, this simple, conservative test has been enough to illustrate the effectiveness of testing for visibility when modeling hair.
Figure 3.13: Visibility of hair is used to choose its representation. (a) User's point of view of the hair (b) Viewpoint from the back to show that hairs the user cannot see are not rendered and are simulated as strips; White lines show the view frustum of the user's camera.

3.6.2 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.

A single strip, a group of c clusters, or a group of s strands represent a given portion of hair. Each is designed to cover a similar amount of world space. By calculating the amount of pixel coverage the hair will have at its current distance, an appropriate LOD can be chosen. The allowable number of pixel error is decided by the user based on their desired error threshold; the number of pixels of error for the system is then 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:

\[ dPP = 0.5 \times \max\left(\frac{wR - wL}{W}, \frac{wT - wB}{H}\right) \]

\[ WSE = \frac{d \times \text{allowedPixelsOfError} \times dPP}{\text{Near}} \]
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 viewing window in pixels. \( \text{Near} \) 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 representation. 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.

### 3.6.3 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 strands or clusters. 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 (strips are thicker than clusters and clusters are thicker than strand groups) the more easily it should break into smaller groups or finer LODs.
3.6.4 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 are therefore typically more viewable, remain at the individual 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 as individual strands instead of clusters, 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. The pseudo-code for the LOD selection process is shown below:

```plaintext
if (occluded == true)
    Simulate as strip
    Do not render
else
    DistanceLOD = TestHairDistance()
    MotionLOD = TestHairMotion()
    // Use whichever test requires highest LOD
    if (DistanceLOD > MotionLOD)
        CurrentLOD = DistanceLOD
    else
        CurrentLOD = MotionLOD
```

3.7 Results and Comparisons

This level-of-detail framework has been implemented to judge its results in C++ using OpenGL for display. The placement of the hair and specification of user parameters described in this chapter were set using an interactive hairstyling tool that was created to allow a user to place the skeletons on the head at desired locations. The styling tool also lets the user set parameters for curly or wavy hair automatically by setting the angles that control the degree
of the curl or wave, respectively. The size (length, width, radius, etc.) of each representation is also set using this styling tool. The user directly controls the placement and size of each strip; the hair system then automatically generates clusters and strand groups from the strips.

In order to speed up the LOD transitions at runtime, many components are pre-computed. The corresponding SSVs for each representation of hair to be used for collision detection are pre-computed. Therefore, during an LOD transition, the only values that need to be updated are the positions and velocities of the skeleton nodes.

### 3.7.1 Performance Comparisons

The system has been tested on various scenarios. The performances for the overall dynamic simulation (not including collision detection) and collision detection using different representations were compared on various simulation scenarios. Table 3.1 gives a detailed comparison of the *average* running times using a combination of LOD representations (indicated as LODs) against the use of only one of the three discrete LOD representations (Strands, Clusters, and Strips). Figure 3.15 shows the runtime comparison of the simulation performance over the entire duration, as the camera zooms out, increasing the distance to the viewer. The rendering performance is similar to that of the simulation and is illustrated in Figure 3.16. The basis for our comparisons uses the average timings for the strand simulation as the value 1 on the graphs.

<table>
<thead>
<tr>
<th>Breakdown</th>
<th>LODs</th>
<th>Strands</th>
<th>Clusters</th>
<th>Strips</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyn Sim</td>
<td>0.0175</td>
<td>0.5834</td>
<td>0.0298</td>
<td>0.00592</td>
</tr>
<tr>
<td>Col Detect</td>
<td>0.0567</td>
<td>2.1934</td>
<td>0.1297</td>
<td>0.01896</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>0.0742</td>
<td>2.7768</td>
<td>0.1595</td>
<td>0.02488</td>
</tr>
</tbody>
</table>

Table 3.1: Performance Comparison. *Simulation for a camera zooming out. The average performance numbers are measured in seconds per frame.*

For this benchmark, 8045 individual strands, which were represented using only 173 strips or 519 clusters, were used. In these benchmarks, a combination of all three discrete LOD representations was automatically determined by the framework at any given time during the simulations. Figure 3.14 illustrates the visual quality of each discrete representation versus
the use of LODs to represent a short, curly hairstyle. Timings were taken on a PC equipped with an Intel® Pentium™ 4 2-GHz processor, 1 GB main memory and an NVIDIA® GeForce™ 4 graphics card.

Strips provide the best overall performance in simulation time, since it is the coarsest (lowest) LOD of hair, but provides the lowest visual quality, as illustrated in Figure 3.14. However, a combination of three discrete LOD representations using this framework offers significant performance advantages over the use of individual strands alone. Note there are also implicitly continuous LOD representations used in the system with the subdivision framework. While it renders images of almost equal visual quality as that of individual strands, our LOD implementation gives much better timing performances than modeling individual strands in simulation, collision detection, as well as rendering. Figure 3.17 shows comparison simulation footage of the LOD framework versus the use of strand groups alone. In this
Figure 3.15: Simulation Performance Comparison. Show the factors of speed-up among LOD, strips, and clusters over the strands alone, which is the baseline for comparison. The simulation speed of the system consistently outperforms the strands. It quickly outperforms the use of clusters alone, as the camera starts to zoom out. Then, soon after a certain distance threshold, it performs comparably to the use of strips alone.

example wind is blowing through the hair and the camera is zooming away from the figure triggering LOD transitions. Note the images at the far right in the figure show the LODs color-coded to illustrate the changing geometric representation of the hair.

3.7.2 Analysis and Discussion

The impetus of this research is to explore the use of level-of-detail representations for modeling hair to automatically generate its aggregate behavior, while preserving the visual fidelity of the overall simulation. It is difficult to meaningfully quantify the computational errors introduced by the use of simplified representations for modeling hair. However, the resulting simulations can be subjectively evaluated by performing comparisons on the visual quality of rendered images, such as that in Figures 3.14 and 3.17
Figure 3.16: Rendering Performance Comparison. Show the factors of speed-up among LOD (in blue), strips (in green) and clusters (in orange) over the strands alone (in red), which is the baseline for comparison. Notice that the rendering speed of the LOD system consistently outperforms the individual strands. It quickly outperforms the use of clusters alone, as the camera starts to zoom out. Then, soon after a certain distance threshold, it even outperforms the use of strips alone due to the automatic occlusion culling used in the switching criteria for rendering acceleration.

Using side-by-side comparison, we notice little degradation in the visual quality of the rendered image using LODs. While they offer the best computational performance, the images of hair simulated by strips appear sharp and angular, lacking a realistic appearance. The performance of this framework varies depending on the scenarios. In general, its overall performance in simulation and rendering compares favorably against the use of strands, clusters or strips alone. However, this LOD approach can automatically place the computing resources towards areas where the hair is most visible to the viewer and, thus, offers a higher visual quality for the resulting simulations in comparison to the use of strips alone [KH01].
Limitations

One possible limitation of this algorithm is the slight popping that can occur if aggressive LOD transitioning is used. However, this visual artifact can be alleviated with motion blurring and image blending during LOD transitions.

Occlusion culling is an active and challenging area of research. To perform occlusion culling for hair rendering presents many more new challenges, as the hair can self-occlude, but each strand, cluster or strip is rather small in size yet in aggregation its capability to occlude the rest of the hair can be significant. The implementation used for these results only has simple object-hair occlusion tests to validate the basic idea. However, there is already a noticeable performance gain using occlusion culling.

Furthermore, in this implementation the three discrete representations were used at static resolutions limiting the possible representation of the hair to only three choices. The next chapter will explain the creation of a hair hierarchy that allows for further level-of-detail control while capturing more intricate dynamic clustering effects.

Comparisons Against Other Approaches

A multiresolution technique for hairstyling was presented in [KN02], which only used groups of strands of varying sizes for static hairstyle creation; furthermore, the switching is directly controlled by the user, while the framework presented in this chapter offers the capability of automatic switching of dynamic hair.

Techniques that use interpolation from guide strands, such as [CJY02], can be limiting in the types of hairstyles that can be modeled as well as its interaction capabilities. Furthermore, these simulations run at slower rates than the LOD framework.

The use of discrete hair LODs compares favorably to those using only cluster-like representations (e.g. wisps or generalized cylinders) [CSDI99, KAT93, XY01, PCP01], as we can achieve similar visual quality with faster rendering and simulation rates.

The real-time animation techniques [KH00, KH01] and those commonly found in video games are either limited to certain hairstyles (mostly short, straight hair) or the resulting
image quality is inferior to those created with this LOD system.

3.8 Summary

In this chapter, I have presented the basic framework for level-of-detail hair modeling. This framework adapts three discrete level-of-detail representations for modeling hair that are created from the base skeleton and subdivision scheme. These parts are the building blocks to the level-of-detail hair modeling framework and play a key role in the simulation and rendering methods described in this chapter. Finally, the system is able to switch between these different representations on the fly, placing most of the computational resources towards the hairs that are most significant to the viewer and to the simulation.

In summary, the key elements of my level-of-detail framework for hair modeling include:

- **The Base Skeleton** provides a unified underlying control structure for each of the three level-of-detail representations. It controls the shape of the hair and provides a simple method for specifying different hairstyles including straight, wavy or curly hair.

- **Discrete Level-of-Detail Representations** for modeling hair, which are designed to be used together to balance between performance and visual quality for hair simulation and rendering. Each LOD has the flexibility to model different styles and motions of hair with varying levels of fidelity.

- **Subdivision Representations** allow for smooth curves and surfaces that can capture the details of the arbitrary shapes of hair.

- **An Implicit Integration** scheme designed for specific hair motion that provides more stability for the simulation, thereby increasing the generality of the system by allowing the use of stiffer springs, while taking large simulation time steps.

- **A Collision Detection** method that efficiently handles hair-hair and hair-object interactions. Through the use of novel *swept sphere volumes* for approximating hair it is possible to create a tight fitting bounding volume for each level-of-detail representation.
Moreover, the SSVs provide a simple and fast intersection check that is applicable for detecting collisions between the hair and the body, as well as among the hairs. Using spatial subdivision further accelerates the collision detection process by locating pairs of SSVs with the highest potential for overlap.

- **A Hair Rendering** approach that uses graphics hardware capabilities to accelerate intricate light scattering and shadow computations. These rendering techniques have also been generalized for each LOD representation and used to alleviate transitioning artifacts.

- **Runtime Selection of Hair** which is based on three criteria for measuring the hair's importance to the application: *visibility*, *viewing distance*, and *hair motion*. These criteria are used together to find the appropriate representation for a given section of hair, thereby accelerating the simulation and rendering of hair that cannot be seen well by the viewer while maintaining high visual quality for hair that desire more detail.

This chapter has shown that the use of the level-of-detail framework can accelerate hair simulation and rendering while keeping the visual quality of the simulation at a level comparable to common wisp-based systems [CSDI99, KAT93, XY01, PCP01]. In the next chapter, a method for generating more control over the level-of-detail framework via a hair hierarchy will be explained along with the technique for LOD transitioning.
Figure 3.17: Comparison simulations of wind blowing through short, wavy hair as the camera zooms out. From left to right (1) Strands (2) LOD Representation (3) LOD Representation color coded: strands are shown in yellow, clusters in red, and strips are mostly occluded but shown in blue when rendered.
Chapter 4

Hair Hierarchy

The previous chapter introduced my basic framework for level-of-detail hair modeling and showed how the three discrete hair representations are used together to accelerate hair simulation and rendering by providing varying resolutions for the performance speed and visual fidelity of the animated hair. In this chapter, I introduce the hair hierarchy, a control structure that provides further refinement to the LOD hair framework. In the previous chapter, a given volume of hair could be modeled as a single strip, $c$ clusters, or $s$ strand groups, where $c$ and $s$ were determined prior to the simulation and remained constant subsequently. The constant number of strips, clusters and strand groups limits the simulation to three choices for the representation of hair. While this technique provides the flexibility to dynamically alter a simulation based on performance speed or visual faithfulness, offering more than three resolution choices increases the flexibility of the framework. The hair hierarchy increases control over the resolution as it contains various numbers and sizes of each discrete representation.

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, nearly 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, including popular wisp-based [WS92, KAT93, DTKT93, PCP02, CCK05] and continuum-based [HMT01, CJY02, BCN03, VMT04] approaches; this effect is the dynamic clustering and splitting of hair strands often exhibited in 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 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 chapter, I explain the construction and storage of the hair hierarchy, which is performed as a pre-process to the simulation. I will then discuss the runtime traversal of the hierarchy, which models the dynamic merging and splitting of groups of hairs as it selects the appropriate resolution LOD for the hair. Finally, I will discuss the results, benefits and limitations of this control structure.

4.1 Construction of Hair Hierarchy

The hair hierarchy uses the continual subdivision of strips, clusters and strand groups in order to attain varying detail based on the current simulation state. The subdivision is
performed as a pre-process and the information is stored in the hierarchy for retrieval during runtime. As the subdivision of a hair group is performed, additional skeletons are added to the system, creating a finer detailed simulation. The hair hierarchy can be viewed as a hierarchy of hierarchies, containing a strip hierarchy, multiple cluster hierarchies, and multiple strand group hierarchies, as illustrated in Figure 4.1.

Figure 4.1: Hair Hierarchy. *One hair hierarchy consists of a single strip hierarchy, multiple cluster hierarchies and multiple strand group hierarchies. The coarsest hair representations are located in the strip hierarchy at the top of the overall hair hierarchy.*

The strand group hierarchy is built in a top-down manner creating smaller groups of strands until the limit of a single strand in a group is reached. Likewise, the hierarchies of clusters are built from top to bottom, as are the strip hierarchies. The strand group
hierarchies are coupled with the cluster and strip hierarchies to attain varying resolutions of each discrete representation. Assumed to be given at the initial setup, the root strip in the strip hierarchy is the coarsest representation for an assemblage of hair, refer to Figure 4.1. In order to gain more detail the strip tree is traversed downward until its leaves are reached. For more detail, a finest LOD strip representation transforms into the coarsest LOD cluster representations, or the root clusters in the cluster hierarchy. Similarly, to attain even more detail, the cluster hierarchy is traversed until the leaves of the cluster tree are reached. At this point, the cluster representations are replaced by the top-level strand groups of the strand group hierarchies. To attain the finest detailed simulation possible, the simulation traverses to the leaves of the strand group trees, which contain individual strands.

The next sections explain the subdivision process and hierarchy building mechanisms starting with the creation of strip and cluster hierarchies.

4.1.1 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. Hairstyle specific information is then declared, defining the length of the hair, the number of control points of the skeleton, and the desired curls or waves down the length of the skeleton (as described in Chapter 3.1).

The hierarchy creation for clusters and strips are more straightforward than the subdivision of the strands, which will be explained in the next section. Strand group subdivision is more complex because of the hair strand geometry which must remain consistent from one level to the next. Clusters and strips represent approximations to the geometry of actual hair strands. The geometric restrictions are not as severe at the cluster and strip level because not as much detail is viewable in comparison to the strand level.

Because the strip is a two-dimensional surface, its subdivision is restricted such that it may only be split into two equal parts, see Figure 4.2. Strip subdivision is simply the degenerate case to cluster or strand group subdivision, using a degenerate quad-tree, or a binary tree,
Figure 4.2: Strip Subdivision. *Strips are split into two equal sized strips.*

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, detailed in Chapter 3.2.2. This circular cross-section is then split into four equal parts. The four sub-clusters have identical radius values 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.
Figure 4.3: 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.

4.1.2 Strand Group Subdivision

A strand group, as explained in Chapter 3.2.3, is defined by a single skeleton, a radius to define the circular cross-section of the group, and the individual strands of hair for rendering purposes. A strand group cross-section is illustrated in Figure 4.3(a). 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 with the line swept spheres used as bounding volumes, explained in Chapter 3.4.2.

A quad-tree data structure contains the hierarchy information, so each strand group is split into four equal sections, as shown in Figure 4.3(b). The subdivision of a strand group into four sections creates a tight fitting circular cross-section for each subgroup, as in Figures 4.3(c) and 4.3(d).

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 4.3(e)). 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 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.

The final strand hierarchy is depicted in Figure 4.4. Each node in the hierarchy contains a strand group, which includes its skeleton and the hair geometry used for the final rendering stage. In addition, each strand group, as well as each cluster and strip, holds its $n-1$ bounding volumes used for collision detection, where $n$ is the number of nodes in the skeleton. The creation of these bounding volumes is described in Chapter 3.4.2.

Each skeleton contains the same number of control points as its parent hair group, which aids in dynamically switching between different levels, to be described next.

4.2 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 Chapter 3.6. The occlusion of the hair is tested first followed by the distance and motion tests. When these tests determine a transition is to occur, the hierarchy either performs adaptive subdivision or adaptive merging of the appropriate hair groups.

Figure 4.5: Adaptive Subdivision: *Two skeletons (left) are dynamically subdivided into multiple (right).*

### 4.2.1 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. As explained in Section 4.1, 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 4.5 shows two skeletons dynamically subdivide into multiple skeletons as a gust of wind blows through the hair.

4.2.2 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; that is, if the criteria explained in Chapter 3.6 for switching levels are satisfied for all of the children.

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 4.6. 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. If this distance for any control point is greater than a certain threshold, the transition will not occur (see Figure 4.6(b)).

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 4.6(c)). If the control points fall within the second threshold, a spring force is used to pull the children into place so a smooth transition may occur (see Figure 4.6(d)).
Figure 4.6: Adaptive Merging. Positional constraints placed on child skeletons merging into parent (a) Parent skeleton (in red) potential position determined by averaging positions 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.

4.3 Results and Comparisons

The hair hierarchy was first presented by Ward and Lin [WL03] and has been implemented and compared with previous hair modeling schemes on various scenarios. In this section, I will review the benefits of the hair hierarchy and discuss the results and limitations of this structure.

4.3.1 Analysis and Discussion

The impetus of the hair hierarchy is to explore the use of dynamic grouping and subdivision of hair to automatically generate nearly continuous LODs for hair simulation. This approach provides further improvement to the visual fidelity of the animated hair while maintaining the accelerated dynamic simulation and rendering presented in the previous chapter. It is difficult to meaningfully quantify the computational errors introduced by the use of simplified representations for modeling hair. Notwithstanding the foregoing, the resulting simulations can be subjectively evaluated by performing comparisons on the visual quality of the simulated
results. Using side-by-side comparisons higher visual fidelity of the simulated hair is noticed using the hair hierarchy in comparison to the three discrete LODs alone. Figure 4.7 compares the static grouping of hair strands with the adaptive grouping technique and shows that the unnatural clumping of hair strands can be alleviated with the hair hierarchy. The performance of the framework varies depending on the scenarios. In general, its overall performance in simulation and rendering compares favorably against the use of discrete LOD representations.

The hair hierarchy offers more continuous control over the simulation than using the discrete LODs alone. The hierarchy provides a more natural progression in detail throughout the simulation due to the varying resolutions of each representation used. Transitions between LODs are then less noticeable to the viewer. Furthermore, a much higher visual quality can be achieved with the use of the hair hierarchy than other hair modeling methods because the hair hierarchy allows the simulation to progress down to the individual strand level, unlike other hair modeling methods. Many wisp-based approaches, though common, suffer from unnatural clumping effects. In contrast, many volume-based methods suffer from the hair moving in an overly continuous manner: they fail to capture strands breaking apart from the rest of the volume.

Figure 4.7: The rendered images without (left) and with (right) adaptive subdivision.

The hair hierarchy alleviates the disturbing clumping behavior often present in other hair simulation techniques described in Chapter 2, while capturing dynamic clustering and splitting effects that occur as strands move from one group to another.
4.3.2 Performance Comparisons

The performance has been compared for the collision detection and dynamic simulation using different representations on various simulation scenarios. The first scenario entails wind blowing through the hair as the camera remains stationary. The camera is positioned close to the figure, so the viewer can see fine detail, and primarily shows the effects of the continuous LODs used within the strand hierarchy. Snapshots of this simulation are displayed in Figure 4.8 using adaptive grouping of hair strands. Table 4.1 gives a detailed comparison of the average running times for this scenario using the adaptive levels of detail from the hair hierarchy (indicated as LODs) against (i) the use of the finest hair representations, or the lowest possible level in the hair hierarchies (indicated as Fine Strands in the table), and (ii) the coarsest strand representations, or highest level within the strand hierarchies (indicated as Coarse Strands in the table). The hair hierarchy allows the simulation of strands with visual detail comparable to that of the finest strand representation (in Figure 4.8 individual strands are clearly shown breaking away from the rest of the hair volume), whereas the timings for the simulations are comparable to that of the coarsest strand representation.

<table>
<thead>
<tr>
<th>Breakdown</th>
<th>Fine Strands</th>
<th>LODs</th>
<th>Coarse Strands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyn Sim</td>
<td>0.107636</td>
<td>0.041624</td>
<td>0.038271</td>
</tr>
<tr>
<td>Col Detect</td>
<td>7.642328</td>
<td>0.411793</td>
<td>0.338298</td>
</tr>
<tr>
<td>Total</td>
<td>7.749964</td>
<td>0.453417</td>
<td>0.376569</td>
</tr>
</tbody>
</table>

Table 4.1: Performance Comparison. Simulation for a stationary camera. The average performance numbers are measured in seconds per frame.

Table 4.2 shows results of the same simulation as the camera zooms away from the figure. With this simulation the influence of the discrete LODs is obvious. The use of clusters and strips with adaptive grouping and subdivision increases the performance of the simulation. The table shows comparisons of the same simulation with the finest detailed strands, versus the coarsest detailed clusters and coarsest detailed strips.

For this benchmark, 9,350 individual strands were rendered. At the finest detail in the hierarchy, these strands were simulated with 3,570 skeletons, averaging 2.6 strands per skeleton. This implementation did not allow all of the hierarchies to extend to each individual...
<table>
<thead>
<tr>
<th>Breakdown</th>
<th>LODs</th>
<th>Strands</th>
<th>Clusters</th>
<th>Strips</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyn Sim</td>
<td>0.026142</td>
<td>0.107636</td>
<td>0.015374</td>
<td>0.003242</td>
</tr>
<tr>
<td>Col Detect</td>
<td>0.239489</td>
<td>7.642328</td>
<td>0.171781</td>
<td>0.020142</td>
</tr>
<tr>
<td>Total</td>
<td>0.265631</td>
<td>7.749964</td>
<td>0.187155</td>
<td>0.023384</td>
</tr>
</tbody>
</table>

Table 4.2: Performance Comparison. *Simulation for a camera zooming out. The average performance numbers are measured in seconds per frame.*

strand due to the overwhelming computational cost entailed. Rather, some hierarchies extend to the individual strand level, while others contain a minimum of four or five strands at the lowest level. This combination enables the simulation to distribute the computational resources where they are needed the most. Hair strands that originate at the top of the head, near the part of the hair, are more viewable and were allowed to extend to the individual strand level, whereas hairs located at the base of the neck are typically not as visible and do not require a hierarchy reaching as far. These 9,350 strands are then represented with 110 strips, at the coarsest level, or 330 clusters at the coarsest level in the cluster hierarchy. The skeletons comprising the hairstyle consist of 6 control points on average. Timings were taken on a PC equipped with an Intel® Pentium™ 4 2-GHz processor, with 1 GB main memory and an NVIDIA® GeForce™ 4 graphics card. The thresholds used to trigger transitions were empirically assigned values for the root (strip) and leaves (strands) of the hair hierarchy. These threshold values were then linearly interpolated to the remaining levels in the hierarchy. A non-linear interpolation scheme could be employed if a different threshold progression were desired.

### 4.3.3 Limitations

As with most LOD algorithms that generate hierarchical representations offline, our approach necessitates considerable memory requirements. Table 4.3 shows approximate memory usage in KB. The benchmark used is the same as stated in the previous section. The “Coarse Strand Groups” entry is comparable to typical wisp implementations. Note that the discrete LODs (referring to the implementation in the previous chapter) consumes more memory than the use of strand groups, or wisps alone, however, the hair hierarchy requires considerably
Figure 4.8: A series of snapshots (a)-(f) showing adaptive grouping of hair strands with wind sporadically blowing through long, straight, brown hair.

more memory than any of the other models. One contribution to this is the storage of the rendered strand geometry, which is copied at each level in the strand hierarchy. Since the number and placement of the strands remains consistent from one level to the next in the strand hierarchy, this information could be stored only once at the leaves of the tree and accessed from there for rendering. Further memory optimizations could be applied to this work for more efficient storage. However, the pre-computation and storage facilitates the accelerated performance during runtime.

Moreover, the hair hierarchy is built upon the assumption that styling information, such
Figure 4.9: Dynamic Simulation of Hair Using LODs. *A sequence of snapshots (from left to right).*

<table>
<thead>
<tr>
<th>Breakdown</th>
<th>Memory Usage (in KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coarse Strips</td>
<td>5,840</td>
</tr>
<tr>
<td>Coarse Clusters</td>
<td>22,860</td>
</tr>
<tr>
<td>Coarse Strand Groups</td>
<td>46,000</td>
</tr>
<tr>
<td>Discrete LODs</td>
<td>75,000</td>
</tr>
<tr>
<td>Hair Hierarchy</td>
<td>215,700</td>
</tr>
</tbody>
</table>

Table 4.3: Memory Comparison. *Approximate memory usage comparisons measured in KB for the coarsest strips, coarsest clusters, coarsest strand groups (typical wisp implementations), discrete LODs (Chapter 3) and hair hierarchy implementation.*

as curl and wave locations, is consistent from one level to the next. While this observation holds for most people’s natural hairstyles, artificial styling can alter the location and size of curls (or other properties) at localized areas so that style information will differ at locations around the hierarchy. Chapter 6.4 explains the methods I have developed for handling the inheritance of special style specific information. The implementation discussed in this chapter is for uniform styles.

In comparison to the use of three discrete representations alone from the previous chapter, the hair hierarchy provides minimal differences for the rendering of animated hair. When the camera is close to the strands so that the simulation is primarily controlled within the strand hierarchies, the rendering performance is not altered because the number of strands rendered is the same from one level to the next. The rendering performance is accelerated slightly through the cluster and strip hierarchies since the number of primitives being rendered decreases towards the root of the hierarchies.
4.3.4 Comparisons Against Other Approaches

As stated before, the adaptive merging and splitting of hair groups is able to capture dynamic clustering and splitting of hair that most wisp-based [WS92, KAT93, DTKT93, PCP02, CCK05] and continuum-based approaches [HMT01, CJY02, BCN03, VMT04] cannot. Moreover, this research is built upon the discrete LOD representations discussed in the previous chapter. Compared to the three discrete representations alone, the hair hierarchy allows for higher quality visual appearances while maintaining similar or better runtime performances. Using the hair hierarchy enables the user to maintain complete control over the visual results and computational performance of the system.

In comparison to the hair hierarchy, a similar multi-resolution control structure was presented by Bertails et al. [BKCN03] concurrently, referred to as the Adaptive Wisp Tree (AWT). Unlike the approach presented here, the AWT only uses strand groups of varying size and does not incorporate the use of clusters or strips for acceleration. Moreover, merging and splitting occur only as a measure of the hair's velocity; distance and occlusion criteria are ignored. The AWT construction is only presented to include up to three varying levels whereas the hair hierarchy is able to extend down to all of the strands due to the subdivision construction presented in Section 4.1. The benefit of the AWT is to model dynamic clustering effects of hair and is not presented to accelerate hair animation in the same manner as the hair hierarchy. The hair hierarchy performs almost an order of magnitude faster than the AWT in simulation and renders at a faster rate using the cluster and strip representations.

The AWT, however, presents a method for splitting a group of hairs starting at the tips of the strands and then working upwards towards the roots. The hair hierarchy automatically splits an entire group of strands from root to tip at once. Our method could be refined to incorporate localized splitting and merging, however transitions between different geometric LODs, such as from strand groups into clusters, would be difficult to handle in this manner since the geometry would be different at varying lengths down the group of hair.
4.4 Summary

In this chapter, I presented the hair hierarchy control structure and transition techniques for adaptively merging and splitting groups of hair. The hair hierarchy provides further level-of-detail control over the hair framework by creating nearly continuous LODs of the hair. Moreover, the hair hierarchy is able to alleviate visually disturbing clumping of hair while capturing adaptive clustering behavior. The key features discussed in this chapter are:

- **The hair hierarchy** which provides the user continuing control over the resolution of simulated hair through the continual subdivision of strips, clusters and strand groups. The hair hierarchy is created automatically requiring the user to specify only a few key parameters.

- **Adaptive splitting** of groups of hair, which can be performed on the fly as simulation conditions change requiring a more detailed animation. Adaptive splitting is performed at any point within the hair hierarchy by moving down a level for extra detail. Splitting applies to a transition from a strip into multiple clusters and a transition from a cluster into multiple strand groups as well as a transition from one geometric LOD to multiple, finer resolutions of the same geometric LOD (one strips transition into two strips).

- **Adaptive merging** of hair groups that is also performed on the fly, simplifying the simulation of hair when less detail is necessary for animating the hair. I presented a method for alleviating visual artifacts that can occur when a dynamics system is simplified by placing constraints on the merging of hair groups.

The techniques discussed in this chapter add to the framework described in Chapter 3 for balancing between the visual fidelity and performance speed of animated hair. The next chapter explains further refinements to these methods for hair modeling through the dynamic application of water and styling products onto the hair.
Chapter 5

Modeling Hair with Water and Styling Products

In the natural world, external substances interact with hair, thereby changing its physical behavior and outward appearance. Water, for example, is often present on hair due to ordinary daily activities such as styling, washing, swimming, and perspiring. In addition, there are numerous cosmetic styling products, such as hairspray, mousse and gel, available on the market that enable consumers to attain desired hairstyles or hair properties. However, existing hair modeling systems, discussed in Chapter 2, have not accounted for these modifications. In this chapter I will illustrate the effects that water and styling products have on hair as well as the method I have developed to capture these key characteristics. I present methods for capturing the essential attributes of hair influenced by water and styling products, allowing for an enhanced hair modeling system that can be used by beauticians, stylists, dermatologists and other physicians to “preview” hair’s appearance and motion under different conditions.

Chapter 3.1 described how the base skeleton model is used to control the geometric shape of each hair representation. Many previous hair simulation algorithms similarly use a single-skeleton system to manage hair dynamics either through hair wisps or guide strands. A single-skeleton system is straightforward to implement and is capable of capturing the basic properties of most hairstyles; however it is limited in a number of ways. Water and styling products change hair in very specific ways. The methods for modeling hair previously have not been developed to anticipate these changes. For this purpose, I have developed a dual-
skeleton system to simulate hair, which is designed to dynamically change various properties of hair when water or styling products are applied to it. The dual-skeleton system offers more flexibility and efficiency in comparison to the traditional single-skeleton system.

This chapter contains a summary of information from biological and chemical literature on hair that explains the effects water and styling products have on real hair. I also describe the dual-skeleton model and show how it controls the motion and structure of the animated hair. Next, I explain the techniques I have developed for dynamically capturing the physical modifications made to hair when water or styling products are applied to it. Moreover, the rendering of hair is also altered on the fly to capture the changes to the visual appearance of hair.

5.1 Background

Before I describe the techniques we have developed for capturing the primary effects of water and styling products on hair, it is important to understand some physical properties of hair fibers and their interaction with these substances.

5.1.1 Hair and Water

Hairs of most mammals, including humans, mainly consist of the protein material α-keratin [Joh97]. α-Keratin is a biological polymer consisting of polypeptide chains, which form with water. Water acts as a plasticizer of the biopolymeric structure of keratin and is able to drastically modify many of the physical properties of α-keratin fibers, including mechanical and electrical properties. These changes are so significant that according to Johnson [Joh97] α-keratin fibers with different water contents can be considered as different keratin-water materials. As a plasticizer, water changes the longitudinal stiffness of fibers by as much as a factor of three as water is fully absorbed into the hair. Furthermore, hair fibers are highly permeable, allowing hair to absorb 30 to 45% of its own weight in water causing the fibers to swell radially by about 16% as the hairs' wetness increases [L'O05, Joh97].

Another easily observable characteristic of wet hair is that it appears flatter than dry
hair. While individual hair fibers swell due to the absorption of water, wet strands in close proximity with each other group together due to the bonding nature of water. As a result, wet hair appears less voluminous than dry hair. Figure 5.1 shows side by side images of a real person with dry and wet hair. Note how much fuller the hair is when it is dry than when it is wet.

![Figure 5.1: Real images of (a) dry and (b) wet hair. Note the difference in the color and hair volume as the strands of hair clump together when wet.](image)

5.1.2 Hair and Styling Products

The purpose of cosmetic styling, or fixative, products is to change the physical properties of hair. A fixative product is designed to take into account the type of hair it will be applied to, the styling techniques it will be used with, and the final results for which it is intended. Styling techniques can include the use of blow-dryers, wet rollers, curling irons, etc. Styling products can be applied in various forms (e.g. glazes, gels, shaping spray, finishing spray, pour-on). For a more comprehensive list of styling techniques, application methods, and ingredients of fixative products, please refer to Johnson [Joh97]. Given the myriad of fixative products on the market, I have chosen to look at the general effects of styling aids on hair.
The functions of styling products are typically to hold a section of hair in place, alter the feel of hair, and/or increase the inter-fibril interactions of hair strands [Joh97]. The application of fixative products prevents fibers from smoothly sliding over each other.

Fixative products usually cause a high degree of adhesiveness in hair, thereby causing hair strands to cling together and move in large groups wherever the product is applied. Polymers constitute the primary active ingredient of most styling products and they increase the stiffness of the hair fibers, thus decreasing the general motion of the hair. Again, exact measurements of these effects vary widely by product. However, the most observable effects of fixative products are typically stiffened hair motion and the bundling of strands [Joh97]. Figure 5.2 shows hair with and without styling products. In this example, hairspray was applied to the hair, causing the strands to adhere to each other.

Figure 5.2: Real images of hair (a) without and (b) with styling products (hairspray). Note how the hair strands cling to each other, decreasing the overall volume of the hair.
5.2 Dual-Skeleton System

Hair motion is subject to changes in the global positioning of the strands, as well as localized styling changes, such as the elongation of a curl under force. While the localized styling motion can be made very stiff through the application of a strong fixative product, the hair is still subject to a global motion when forces are applied to it.

Single-skeleton dynamics systems, such as the one presented in Chapter 3.4.1, cannot easily capture the separation of these two features. The global movement of hairs and the local changes in style are both controlled through the same angular torque equations shown in Equations 3.3 and 3.4. Therefore, simulating a curly section of hair undergoing a large global displacement while retaining a tight curl would be difficult to control, see Figure 5.3.

![Figure 5.3: A curly section of hair can undergo the same global movement, but have differing local motions (a) tight curl is retained (b) loose curl is elongated under force.](image)

An alternative single-skeleton scheme is to use a spring-mass system, such as those used by [RCT91, PCP02, BKCN03]. The spring-mass approach creates a single skeleton curve modeled as a set of particles connected with stiff springs and hinges. Wavy hairs are produced by specifying the number of waves and amplitude of waves inside each wisp. As the wisp segment stretches, the amplitude and frequency of the waves are adjusted to show the wavy hair stretching straight.

The single-skeleton dynamics of the spring-mass system can capture the deforming hairstyle; however, there are no checks to ensure that the length of the hair is preserved over time or that the collision detection remains both accurate and efficient throughout the simulation in light
of the changing orientation and position of the hair inside of the wisp. Wisp-based methods rely on either a deformable bounding volume for collision detection [PCP02], which is expensive to update and to detect collisions, or static cylinder bounding volumes (BV) [BCKN03], which provide a fast intersection test but may not provide a tight, accurate bounding volume fit in light of the changing orientation and position of the hair inside of the wisp.

In order to capture both the global motion and localized styling motion of hair, to ensure hair length preservation at all time, and to maintain an efficient collision detection method throughout the simulation, I have created a dual-skeleton system for modeling hair. This dual-skeleton system provides a single skeleton to control the global motion of hair and a second one to provide a positioning guide and localized collision detection scheme for hair; these skeletons are referred to as the global-skeleton and the local-skeleton, respectively. The preliminary version of this work was presented by Ward et al. [WGL04].

The dual-skeleton system can effectively model all common behaviors of hair, but it is particularly well-suited to modeling the influence of external substances on hair, such as water or styling products. These external substances are often designed to change the physical behavior of hair in specific ways, creating a desire for a special control structure to model hair that anticipates this varying structure. The dual-skeleton system is better suited to model these kinds of changes than previous hair modeling systems because it provides specific control over multiple hair properties including stiffness of bending, stiffness of curl elongation, hair mass, and hair thickness, all of which can be changed at any point along the dual-skeleton.

5.2.1 Dual-Skeleton Setup

The global-skeleton is modeled as a series of node points, \( N_{g0}, N_{g1}, \ldots, N_{g(n-1)} \), where \( n \) is the number of node points, which are connected with springs. A hairstyle is defined by positioning the local-skeleton in the desired form in relation to the global-skeleton, see Figure 5.4. Let the line segment between the \( N_{gi} \) and \( N_{g(i-1)} \) global-skeleton node points be the \( i \)th global-skeleton segment, \( S_{gi} \). The \( i \)th local-skeleton node, \( N_{hi} \), lies in the plane perpendicular to \( S_{gi} \) containing \( N_{gi} \). Each local-skeleton node has a defined angular position to fix it around the global-skeleton segment.
The rendered hair geometry follows the form of the local-skeleton. Chapter 3.1 explained the use of the base skeleton for positioning each LOD representation. Here, the base skeleton corresponds to the local-skeleton because the hair geometry always adheres to its form. Ringlet curls, for example, are simulated using a straight global-skeleton and a local-skeleton that coils around the global-skeleton. Additional hairstyles can be created by forming the local-skeleton in any position relative to the global-skeleton. Curls or waves can be placed at any point down the length of the skeletons and the styles do not need to be limited to cyclic forms, such as ringlets. For straight hair the position of the local-skeleton is equal to that of the global-skeleton. The dual-skeleton is able to model any hair shape or style that can be created with a single-skeleton system; moreover, it offers greater flexibility in dynamics than the single-skeleton system.
5.2.2 Dual-Skeleton Dynamics

In the dual-skeleton system, the global-skeleton is responsible for all dynamics on the hair and is controlled by two types of motion. The first dictates the bending of the strands by maintaining a spring force to control the angular position of each node point in relation to its neighbors. This motion follows the same equations outlined in Chapter 3.4.1 for single-skeleton dynamics. The additional motion of the global-skeleton is the elongation it experiences between two consecutive nodes. Unlike the single-skeleton system described in Chapter 3.4.1, which is comprised of rigid links connecting two consecutive nodes, the global-skeleton connects nodes with springs that allow the global-skeleton to stretch and compress in length. In contrast, the local-skeleton is made up of rigid links and its length always remains constant. Each global-skeleton section is prevented from stretching beyond the reach of its corresponding local-skeleton section. For straight hair the nodes of the global-skeleton are rigidly connected and the position of the local-skeleton is equal to that of the global-skeleton.

The stretching capability allows the dual-skeleton to model the elongation of curly or wavy hair, while the rigidity of the local-skeleton ensures hair length preservation throughout the simulation. As the global-skeleton stretches, the local-skeleton moves closer to the exact position of the global-skeleton. The distance of the local-skeleton from the global-skeleton is thus dependent on the current length of the global-skeleton.

The global-skeleton is responsible for the motion of the hair at any given time. All forces are applied to the global-skeleton and it, in turn, controls the bending of the strands as well as the stretching and compression of the curls. The local-skeleton is not directly related to the physical motion of the hair, but instead plays a vital role in the positioning of the hair geometry and in collision detection, which will be explained in the next section.

The only value to compute for updating the position of the local-skeleton node is its distance from its corresponding global-skeleton node point (see Figure 5.4). First, the current distance between $N_{gi}$ and $N_{g(i-1)}$ is calculated, $d_{gi} = ||S_{gi}||$. The root node, $N_{g0}$, of the local-skeleton is set equal to the root node of the global-skeleton. The distance between $N_{li}$ and $N_{l(i-1)}$ of the local-skeleton is fixed, $d_{li}$. To calculate the position of local node $N_{li}$,
distance \( d_1 \) is calculated by:

\[
d_1 = \sqrt{d_{f1}^2 - d_{g1}^2}
\]

The remaining local-skeleton nodes are calculated using:

\[
d_i = \sqrt{(d_{i-1}^2 + d_{f1}^2) - d_{g1}^2}
\]

Thus, as the global-skeleton elongates, increasing \( d_{g1} \), the local-skeleton straightens and the node points of the local-skeleton move closer to the global-skeleton, decreasing \( d_i \). Naturally, as the global-skeleton contracts, the local-skeleton node points move farther away from the global-skeleton.

5.2.3 Localized Collision Detection

The collision handling technique follows the same spirit as that presented in Chapter 3.4.2, which introduced the use of swept sphere volumes (SSVs) as hair bounding volumes. Applying the SSVs to the dual-skeleton results in a more efficient collision detection system than other spring-mass skeleton systems, such as [PCP01, BKCN03], due to the localized placement of the SSVs. The benefits of using SSVs as bounding volumes are the tight fit it provides to the hair geometry and the simple collision detection test. The previous single-skeleton system I explained in Chapter 3 positioned the SSV so that the rigid line segments between control points were used as the core skeleton of the SSV, a natural method since this rigid line segment is also at the core of the hair geometry. Meanwhile, for single-skeleton spring mass systems the skeleton is no longer guaranteed to be at the core of the rendered geometry, especially for non-straight hair [PCP02, BKCN03]. The bottom left image of Figure 5.5 illustrates the loose fit that occurs when a bounding volume is placed around a single-skeleton spring mass system.

In the dual-skeleton system, the local-skeleton is used for collision detection since the rendered strands adhere to its motion and because the local-skeleton is always at the core of the rendered hair geometry. The radii of the cross-sections at each local-skeleton node point
define the offset for the SSVs, similar to the construction of SSVs in the single-skeleton system. Figure 5.5 illustrates the use of the local-skeleton for SSV placement. Note how the localized placement of the LSS provides a tighter fitting bounding volume than the single-skeleton can provide.

![Diagram of Dual-Skeleton local placement of SSV and Single-Skeleton global placement of SSV]

Figure 5.5: Localized placement of a line swept sphere. *Note that the local placement (shown in red, top) provides a much tighter fit in comparison to previous techniques that place a bounding volume based on a single-skeleton (shown in grey, bottom). (Right) illustrates an entire group of strands with multiple SSVs.*

When a collision is detected between the hair and the body, the global-skeleton of the hair section is moved so that the local-skeleton (and hair geometry) are outside of the body. All positional changes are first made on the global-skeleton and then the local-skeleton follows the position update formulation described in the previous section. It is important to note that if the global-skeleton collides with an object, but the SSVs of the hair, and the corresponding local-skeleton, do not collide with the object, then no action is taken. If a collision occurs, the velocity of the global-skeleton in the direction of the object is set to zero and a frictional
force is applied.

This localized collision detection method allows the local-skeleton to be positioned farther away from the global-skeleton, increasing \( d_1 \), while maintaining tight fitting bounding volumes. This approach results in creating full, voluminous hair.

5.3 Modifications of Physical Properties due to Water and Styling Products

The dual-skeleton is appealing in that it can capture far more properties of hair than previous techniques since it is able to account for various physical changes that occur when hair absorbs water or when fixative products are applied. Section 5.1 summarized the primary influences water and styling products have on hair. In this section, I will explain the methods I have developed for modeling these main physical characteristics, which include:

- **Adjusting dynamics properties** to account for changing mass and spring stiffness of the hairs *on the fly*;

- **Flexible geometric structure** to allow the hair volume to change due to the presence of external substances;

- **Bonds between strands** that form and break *dynamically* reflecting the connection of strands due to water or styling products.

5.3.1 Adjustment of Dynamics Properties

The first two physical changes of hair the dual-skeleton accounts for are the changes in mass and spring stiffness. As water is absorbed into hair fibers the mass of the hair increases up to 45% and the stiffness of the hair fibers increases by a factor of three. As styling products are applied to hair, the stiffness of the hair is increased and the mass of the hair is also increased due to the presence of additional substances. However, it is difficult to put exact numbers on the amount the stiffness and the mass change from styling products since
these real products can vary drastically. Instead, I present a general method for increasing spring stiffness and mass of hair fibers on the fly for dynamic simulation.

As described in Section 5.2, the global-skeleton consists of nodes that control the motion of the hair strands. External forces, such as wind and gravity, are applied to the node points following the standard equation of force:

$$F_i = m_i a_i. \quad (5.1)$$

$F_i$ is the force applied to the $i$th node, $m_i$ and $a_i$ are the mass and acceleration of the $i$th node, respectively. The mass of the nodes are varied to correspond to the length of the hair representing the non-uniform weight of strands from the root of a strand to its tip. Typically the hair has more weight pulling on it at the roots than at the tips.

As the fraction of wetness of the hair, $f_{wetness}$, increases to 100%, the mass of the hair increases up to 45% of its initial dry weight. The mass of the $i$th node, $m_i$, is then calculated as:

$$m_i = m_{dryi} (1 + 0.45 \times f_{wetness}) \quad (5.2)$$

where $m_{dryi}$ is the initial, dry mass of the $i$th node. Using this formulation it is straightforward to add water to specific control points without affecting others.

The internal forces acting on each node point $i$ consist of angular torque, $M_{\theta i}$ and $M_{\phi i}$ for the $\theta$ and $\phi$ components from Equations 3.3 and 3.4, as well as the spring force controlling the length of each global-skeleton section, $F_{leni}$. All of these internal forces are spring forces containing separate spring constants, $k_{\theta i}$, $k_{\phi i}$, $k_{leni}$, respectively. The final spring force equations on the global-skeleton become:

$$M_{\theta i} = -k_{\theta i}(\theta_i - \theta_{i0}) \quad (5.3)$$

$$M_{\phi i} = -k_{\phi i}(\phi_i - \phi_{i0}) \quad (5.4)$$

$$F_{leni} = -k_{leni}(d_{gi} - d_{gi0}) \quad (5.5)$$
where $\theta_i$, $\phi_i$, $\theta_{i0}$, $\phi_{i0}$ are the current angle values and resting angles of the $i$th node in polar coordinates, respectively, and $d_{gi}$ and $d_{gi0}$ are the current and resting lengths of the $i$th segment of the global-skeleton, respectively. With a high, or stiff, $k_{ten}$ value, the hair will be able to bend freely in the $\theta$ and $\phi$ directions, but will not stretch or compress as liberally. The $F_{ten}$ force controls the elongation of the global-skeleton and, hence, the local-styling motion of the dual-skeleton system.

As styling products are applied to the hair, the spring stiffness constant $k_{ten}$ is increased. While the $k_\theta$ and $k_\phi$ values are increased slightly to model the stiffened overall hair motion, increasing the $k_{ten}$ value provides the most observable cue that styling products have been applied to the hair because it controls the local-styling motion. The amount to increase the stiffness depends on the product that is being used. Using the implicit integration technique explained in Chapter 3.4.1, the spring stiffness constants can be increased by a factor of 10 and maintain a stable simulation. Furthermore, this increase is enough to see a visual difference in the motion of the hair so that it resembles fixative product results, see Section 5.5 for comparison images and a discussion of the results.

### 5.3.2 Flexible Geometric Structure

As explained earlier, as hair gets wet it becomes less voluminous. The wet strands bond together flattening out the overall hairstyle. To account for this property, when water is applied to the hair, the thickness of the hair sections decrease accordingly. The radius contraction, or width contraction for strips, becomes a linear relationship based on the percentage of water absorbed into the hair and the number of hair strands within the group of hair. For a cluster or a strip, the number of strands it approximates, meaning the number of strands it would eventually transition into, is used for this value. There is a limit to the amount of water hair can absorb. Moreover, there is a limit to the amount the radius can shrink. The radius can only shrink to the size of all of the strands nestled next to each other. At 100% wetness, the thickness of a group of hair will be equal to the number of strands times the thickness of a strand:
\[ \text{AreaOfGroup} = (\text{AreaOfStrand}) \times n \]
\[ \pi R_{wi}^2 = (\pi r^2) \times n \]
\[ R_{wi} = \sqrt{n} \times r \]  

(5.6)

where \( R_{wi} \) is the radius of the \( i \)th cross-section at 100% wetness, \( n \) is the number of strands in the current strand group, cluster, or strip approximation, and \( r \) is the radius of a single strand. While a single strand can swell due to the absorption of water, the difference between the width of a dry strand and that of a wet strand is typically so minute (approximately 16 \( \mu \)m) that the visual difference need not be reflected in the system, especially in comparison to the overall hair volume contraction due to water absorption.

This radius contraction is then extended to variable amounts of water by creating a linear relationship based on the amount of wetness in the system:

\[ R_{ci} = R_{di} - (R_{di} - R_{wi}) f_{wetness} \]  

(5.7)

where \( R_{ci} \) is the current radius of the \( i \)th cross-section, \( R_{di} \) is the dry radius of the \( i \)th cross-section, and \( f_{wetness} \) is the fraction of water absorbed at the given time.

Moreover, the number of strands can also vary at each level, or cross-section, of the strand grouping. Not all strands of hair are exactly the same length, so the system reflects this observation by varying the radius at each level of the strand grouping, cluster, or strip. These effects are illustrated in Figure 5.6, which shows a section of curly hair at 0\%, 50\%, and 100\% wetness using the strands representation. Note also how the curls begin to elongate due to the extra mass on the hair from the water.

As the thickness of the hair groups fluctuate, the offsets of the SSVs used for collision detection are automatically updated reflecting the change. The offset used to compute each SSV is modified to the new radius of that particular section. This process is performed on-the-fly allowing the radii of the hair representation to change dynamically.

Furthermore, many styling products can be modeled using this flexible geometric repre-
sentation as well. In particular, the purpose of many cosmetic products is to make the hair appear thicker. For example, a “volumizing” mousse, similar to this technique, obviously does not add more hair to a person’s scalp, but the result is to make the hair appear fuller. Therefore, this technique to alter the hair’s volume on the fly is applicable for modeling the effects of these cosmetic products. The process has been presented, thus far, to decrease the volume of the hair, but it can be altered to add volume by changing the parameters:

\[
R_{ci} = R_{di} + (R_{maxi} - R_{di})f_{styling}
\]

where \( f_{styling} \) controls the amount the volume will change, \( R_{maxi} \) is the maximum value the radius can attain, \( R_{wi} < R_{di} < R_{maxi} \) and \( 0 < f_{styling} < 1 \).
5.3.3 Dynamic Bonds between Strands

Due to the bonding effects of most fixative products, hair strands tend to adhere to each other where the product has been applied. I have extended the use of the static links exercised by Chang et al. [CJY02], which were used as breakable connections between guide strands to enable hairstyle recovery. In [CJY02], these links were selected and set up prior to the simulation and were broken when they encountered excessive forces. Here, “dynamic bonds” that model bonding forces between sections of hair are created on-the-fly when fixative products are applied. They can be created at any point in a simulation at any place along the dual-skeletons.

The dynamic bonds are modeled as spring forces connecting two separate nodes of nearby global-skeletons. The hair-hair collision detection method, described in Chapter 3.4.2, identifies which sections of hair are touching. When a styling product is applied to the hair, each section of hair maintains a list of the hair sections with which it is in contact. The dynamic bonds are then formed connecting the corresponding sections. A single section can have as many bonds as hair sections it is touching. The new bonding spring force, $f_{\text{bond}}$, between two global-skeleton nodes becomes:

$$f_{\text{bond}} = -k_{\text{bond}}(d_{\text{current}} - d_{\text{initial}})$$

(5.8)

where $k_{\text{bond}}$ is the spring constant of the bond, $d_{\text{current}}$ is the current distance between the nodes and $d_{\text{initial}}$ is the distance between the two nodes when the fixative product is first applied.

Following methods similar to [CJY02], these bonds are broken when a large force is applied to it. The force required to break the bond is directly related to the strength of the spring constant $k_{\text{bond}}$, which is determined based on the amount and strength of the fixative product applied to the hair.

These dynamically forming and breaking bonds cause large sections of hair to group together and move in union, reflecting the adhesive effect of real fixative products. Moreover, these bonds restrict the hair’s ability to move over or past each other, an effect exhibited in
real hairs due to the increased frictional force caused by fixative products on hair [Joh97].

5.4 Rendering Wet Hair

The rendering of wet materials is still an ongoing topic of research; however it has been observed that water can make objects look darker, brighter, and/or more specular [JLD99]. Jensen et al. [JLD99] presented an approach for rendering wet objects by combining two theories in optics regarding the appearance of wet materials. The first considers the layer of water that forms on the surface of an object and the second takes into account the water inside, below the surface of the object. The combination of liquid on the surface and inside of an object changes the way light rays interact with it. The resulting darker, brighter, more specular appearance is also demonstrated on wet hair strands, as illustrated in Figure 5.1.

To model the appearance of wet hair, we have developed a method that combines the technique of Jensen et al. [JLD99] to model wet materials with the technique of Marschner et al. [MJC+03] to model light scattering from (dry) hair fibers. Our technique takes into consideration the general structure of hair fibers, the thin film of water that forms around hair fibers as well as the absorption of water into the fibers.

5.4.1 Interaction of Light with Wet Hair

When hair becomes wet, a thin film of water is formed around the fibers. The rough, tiled air-fiber interface, observed by Marschner et al. [MJC+03], changes to a smooth, mirror-like air-water interface, see Figure 5.7. This change naturally gives the hair a shinier appearance due to specular reflections, typically modeled by an increasing fall-off exponent in the Phong illumination shading model.

Furthermore, some light rays that penetrate the water surface get trapped within the thin film of water by a phenomenon called total internal reflection, governed by Fresnel’s equations [BW97] for dielectric media. This phenomenon is the first cause for wet hair to appear darker than dry hair [JLD99].

Although the formation of a water film can cause the strands to grow radially by up to
16% of its original radius, their width typically takes up less than a pixel in screen-space. Therefore, we have opted not to change the thickness of a wet strand. On the other hand, the water film does influence the eccentricity of the hair fibers, resulting in a smaller variation of the observed color between individual hair strands.

Due to water absorbed inside and surrounding the hair fibers, the relative index of refraction decreases. The absorption of light inside the hair then increases due to a greater amount of total internal reflection. This phenomenon causes a darker appearance of wet hair in comparison to dry hair. The absorption of water increases the opacity value of the hair and leads to more aggressive self-shadowing.

Moreover, the increased absorption of light due to total internal reflection implies that a smaller fraction of light that is radiated by the hair has traversed the fiber core. Hair fibers have a pigmented core at the center of the cylindrical volume, the source of its apparent
color. Therefore, only the fraction of light rays that traverse the hair core are responsible for the color observed [MJC+03]. Consequently, with increasing wetness, the hair radiates an increasing fraction of colorless light, originated at the air-water surface reflections. This effect contributes to the shinier appearance of wet hair over dry.

5.4.2 Capturing the Rendering Influences of Water

The changes induced by water particles on the surface and inside the hair fibers are simulated by applying a qualitative mapping to three parameters in the shading algorithm. The three main parameters in the shading algorithm are the opacity value $\alpha$, the shininess $s$ and the contribution of the anisotropic lighting fraction $f_a$. Following the rendering algorithm described in Chapter 3.5, these parameters are applied in different sub-procedures of the algorithm. More specifically, $\alpha$ controls the opacity shadow map [KN01] for producing shadowing effects on the hair, $s$ is the exponent for the specular reflection term, and $f_a$ determines the contribution of (partly) non-colored anisotropic reflection. The rendering equation for each hair strand can be expressed as:

$$I_o = (1 - f_a)k_d I_d + f_a(k_d(L, N') + (k_1^1(V, R_1)^{s_1} + k_2^2(V, R_2)^{s_2}))I_i$$

where $L$ is the light direction, $N'$ is the projection of the light vector $L$ onto the normal plane [HS98], and $R_1$ and $R_2$ are the reflected directions for the two specular terms observed by [MJC+03]. The diffuse color $k_d I_d$ is simply the strand color, while the incoming light from the scene is $I_i$. The shininess value $s$ is broken into $s_1$ and $s_2$ for the two separate specular terms.

The interactions of light with the wet strands have been captured by varying the rendering parameters based on the amount of water present on the hair. More specifically, the parameter vector $V_p = [\alpha, s_1, s_2, f_a]$ is varied based on the amount of water on the hair. Extreme values are empirically obtained, defined by $V_p^{min}$ and $V_p^{max}$. The following formula controls the linear interpolation of the values based on the wetness percentage $f_{wetness}$:
\[ V_p = V_p^{\min} + f_{\text{wetness}}(V_p^{\max} - V_p^{\min}) \] (5.10)

As the wetness factor varies between 0% and 100%, the parameters vary accordingly, creating a damp or wet look for the hair strands.

### 5.4.3 Results of Wet Hair

The results of the rendering algorithm discussed here conform with the qualitative observations in [JLD99, MJC+03], showing a large dependence on the \( f_a \) parameter. Indeed, the increased contribution of the anisotropic reflection is consistent with their observations that subsurface scattering of light in the water-penetrated material is the main reason for materials to appear wet. In the case of hair strands, light is subject to total internal reflection in the thin water film on the strand's surface as well as inside the cylindrical volume, causing the darker appearance. Consequently, the colorless air-water surface reflections contribute more to the final shading value, causing the considerable increase in shininess, as expected for typical wet hair.

Although our approximate approach using graphics hardware achieves somewhat limited realism, we believe the visual result is still satisfying for interactive applications. Moreover, this technique can be incorporated into a more realistic ray-tracing model for producing a photorealistic image as [JLD99, MJC+03] performed for similar computations. However, the computation speed of such methods is far too overwhelming for applications that desire interactivity; ray-tracing a single frame of a hair model using the shading algorithm presented by [MJC+03] took 8 minutes.

### 5.4.4 Influence of Styling Products

Cosmetic styling products are comprised largely of water or alcohol. As a result, fixative products alter the appearance of hair in a way similar to water. Though a subtle effect, Figure 5.2 shows hair with hairspray to be slightly darker than the hair without it. The rendering system presented here accounts for these changes by varying the lighting parameters based
on the ingredients and desired effect of the product being applied. For example, a styling product that contains 20% water is then rendered with 20% wetness ($f_{wetness}$).

5.5 Results and Discussion

This chapter has explained a few straightforward methods for capturing the influences of external substances on hair. In this section, I will discuss the results and review a few limitations of the work.

5.5.1 Analysis

Water and styling products change the physical properties of hair, which then alter its dynamic motion. The results of these techniques are illustrated in Figures 5.8 - 5.10. Figure 5.8 shows, from left to right, the same short, curly, blonde hair model wet, dry, and with styling products. The styling product model in this figure it meant to emulate the effects of hairspray, note the tight curls created in comparison to the normal, dry image.

Figure 5.9 shows similar effects on long, curly red hair blowing in the wind. Note the limited overall motion of the wet hair in comparison to the dry hair as well as the elongation of the curls due to the weight of the water. Moreover, the hair with styling products (hairspray) applied moves in large groups due to the dynamic bonds and the curls are tightly retained from the increased spring constants in comparison to the dry, normal hair.

Furthermore, Figure 5.10 shows a sequence of snapshots from animation footage of wind blowing through long, straight brown hair dry (top) and wet (bottom). In the sequence, from left to right, the wind force is decreased until the hair comes back to rest. In these images, note the difference in motion of the wet hair versus the dry hair and the rendering effects as the wet hair appears darker and shinier than the dry hair.

These images demonstrate that the techniques I have developed can capture many intricacies of hair influenced by external substances such as water or hairspray. Using the properties of water and hair fibers as well as the characteristics of cosmetic styling products, this system can effectively model the key features exhibited by hair when in contact with these substances,
as shown in Figures 5.1 and 5.2. As water is absorbed into the hair, the mass of the hair fibers increases, the hair becomes less voluminous, and the longitudinal stiffness is altered (by up to a factor of three). When styling products are applied, the mass and spring stiffness are changed, the volume of the hair can be increased or decreased depending on the product, and dynamic bonds are added between sections of hair to simulate the adhesive qualities of most fixative products.

This system requires no pre-computations to dynamically add water or styling products to the hair, nor does it require any additional calculations once the substances have been applied to the hair. Therefore, the simulation runs at approximately the same rate with or without external substances present.

On average the results took 4.16 seconds per frame for simulation and 0.34 seconds per frame for rendering for a hairstyle with 16 nodes per strand and a total of 9,680 rendered strands. Timings were taken on a PC with a 1.8 GHz processor, 1 GB RAM, and an NVIDIA® GeForce™ 4 graphics card. In the examples shown in this chapter, no levels-of-detail were used to model the hair and all simulations and renderings were performed on a constant number of strand groups. The next chapter will explain the integration of water and styling products with the LOD framework for the purpose of interactive hairstyling.
5.5.2 Limitations

The implementation discussed in this chapter applied an even amount of water or styling products to all the hair in the application. This chapter discussed the process for capturing the basic results of these substances. Modeling these effects on local areas of the hair create interesting results, which will be discussed in the next chapter.

The system presented here could also be advanced by using exact measurements of different styling products. Using precise measurements would be interesting to compare the results of different styling products and would be of more direct benefit to the cosmetics industry as it could be used for virtual testing of their products. Moreover, there are more properties that can be included into the wet hair simulation, such as modeling the plasticity of wet hair. The methods presented in this chapter are meant to simulate the key characteristics of wet hair and hair with styling products to demonstrate processes for dynamically changing the
physical structure, motion and visual properties of animated hair.

5.6 Summary

This chapter has explained the methods developed for capturing the primary effects of water and styling products on hair. The preliminary results were published by Ward et al. [WGL04]. The techniques developed here can dynamically change the properties of the hair so that water and styling products can actually be applied to hair on-the-fly. The key factors discussed in this chapter include:
• The **dual-skeleton** system that decouples the control of the global and local motions of hair. The dual-skeleton is then able to capture multiple properties of hair while keeping a localized, efficient collision detection scheme; it is particularly suited for modeling changing hair properties caused by water or styling products.

• **Automatic adjustment of dynamics properties** to model changing mass and spring stiffness of hair as substances are applied to it.

• **Flexible geometric structure** that captures the fluctuating volume of hair, which is particularly important for modeling wet hair.

• **Dynamic bonds** that are placed between sections of hair in contact with each other when styling products are applied to the hair. The dynamic bonds model the adhesiveness between strands as well as prevent hairs from smoothly passing over each other caused by the application of most styling products.

• **Wet hair rendering** which is modeled by parameterizing the key lighting values influenced by the application of water. As hair becomes wet, there is a noticeable increase in the colorless specular reflections and the hair grows darker.

The next chapter explains the incorporation of water and styling product effects into the level-of-detail hair framework as well as methods I have developed for allowing a user to dynamically place these substances onto the hair. Together these routines are used to create an interactive hairstyling system.
Chapter 6

Interactive Hairstyling

User interaction with animated hair is useful for various applications. For example, virtual environments depicting human avatars require a hair modeling system capable of animating and rendering hair at interactive rates. Due to the performance requirement, many interactive hair modeling algorithms tend to lack important, complex features of hair, including hair interactions, dynamic clustering of hair strands and intricate self-shadowing effects. Often, the animated hair’s appearance and behavior are compromised for real-time interaction with hair.

Because the shape of the hair model and other properties are typically dictated by hairstyling, it is an important step to modeling hair, as discussed in Chapter 2. In order for the virtual hairstyling processes to resemble real-world styling processes both high performance simulation and realistic rendering are required to allow interactive use and the incorporation of fine details. Hairstyles result from physical properties of hair and hair mutual interactions. Thus, hair dynamics must be incorporated to mimic the process of real-world hairstyle creation. Moreover, in the real world people are accustomed to hairstyling by touching hair directly. An intuitive virtual hairstyling tool has to incorporate user interaction with dynamic hair. Until recently, the complexity of animating and rendering hair had been too high to accurately model all of these essential features at desired rates that would allow a user to interact and style virtual animated hair. 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 chapter, I present a physically-based virtual hairstyling system, referred to as the virtual hair salon, that mimics real-world hairstyling processes and requires its users have no prior knowledge other than common hair manipulation techniques. This hairstyling method combines the level-of-detail framework (Chapter 3), hair hierarchy (Chapter 4), and water and styling product effects (Chapter 5), which were all described earlier in this dissertation. By using multi-resolution simulation techniques and graphics hardware rendering acceleration, a physically-based virtual hair salon system is created that animates and renders hair at accelerated rates, allowing users to interactively style virtual hair in a natural manner. With an intuitive 3D interface, users can directly manipulate and position hair strands, as well as employ real-world styling applications (e.g. cutting, wetting, applying styling products) to create hairstyles as they would in the physical world.

This chapter presents an intuitive 3D interface developed for interactive hair manipulation. The interface attains 3D input from the user through a PHANToM stylus, which allows the user to perform several real-world applications on hair, such as cutting or wetting. I
also introduce a simulation localization technique that is integrated into the level-of-detail framework for further acceleration of the modeled hair. This technique regulates the dynamic simulation of the hair based on the motion of the hair and the user's interactions, providing an additional criterion for determining the appropriate resolution to animate the hair.

Moreover, in this chapter I discuss the implementation of several common hair salon applications that are used to change the structure, behavior, and/or appearance of the hair on-the-fly. The supported applications include wetting, cutting, blow-drying, grabbing and moving, applying hairspray and applying mousse to the hair. These applications represent an array of real-world procedures that are commonly performed on hair and are used to demonstrate the framework's ability to effectively use levels-of-detail to allow a user to style dynamic hair. The flexibility and effectiveness of this system is demonstrated through several hairstyles created such as in Figure 6.1, which was created in less than 10 minutes. This work was originally introduced by Ward et al. [WGL05].

6.1 User Interface

It is important to have an intuitive user interface that allows a user to work with the system with little training or requirement of outside knowledge. Many other virtual hairstyling methods require its users have an understanding of complex physical or mathematical formulations, such as fluid dynamics [HMT00]. Meanwhile, in the real world there are many applications performed on hair that small children are capable of performing (e.g. wetting hair). The virtual hair salon system described here uses a SensAble Technologies' PHANToM as a 3D user input device. The real-time display of the PHANToM input is rendered using a commercial haptic toolkit called GHOST. The position and orientation of the device are updated and rendered at each frame. Figure 6.2 illustrates a user operating the system with the PHANToM stylus.

The virtual hair salon is designed to allow its user to switch between virtual tools (e.g. scissors, hairdryer) using the PHANToM stylus in a seamless manner. For this purpose, a 2D menu is projected onto the 3D scene containing the avatar and hair model. The 2D menu
contains graphical icons depicting each tool available to the user. The user interacts with both the 3D scene and the 2D menu using the PHANToM stylus in a seamless fashion. By positioning the stylus over a 2D menu icon and pushing the stylus button, the user chooses a desired application. The position, orientation, and area of influence (the space the application will affect) of the current application is depicted in the scene with a semi-transparent cone or triangle (depending on the application). As the user moves the stylus, the area of influence interactively follows the position and orientation of the user's hand in 3D. The camera is controlled through simple navigation of the mouse.

Figure 6.2: User Interface PHANToM provides 3D user input and 2D menu buttons are labeled with icons to show applications.

6.2 Interactive Dynamic Simulation

Hairstyling in the natural world is performed by focusing on specific sections of hair and executing a desired task on the localized region. This focus correlates naturally with the use of multi-resolution techniques to simulate hair by simulating the areas of high focus with more detail. The level-of-detail hair framework described previously in this dissertation is able to automatically determine the areas of the hair volume that are most significant to the simulation based on the hair's visibility, viewing distance and motion. Now the user's interaction with the hair defines a new criterion for measuring a hair section's significance.
The areas of the hair that the user is directly manipulating (such as by cutting or wetting) is naturally going to draw the user's focus. Moreover, the manipulation of the user will change the hair's physical structure, behavior, and/or visual appearance, implying that more computational resources should be allocated towards these areas to capture such detail.

In this section, I introduce a simulation localization technique based on spatial decomposition that is used to rapidly locate the areas of highest activity. These areas are primarily based on the user's interaction with the hair and 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.

6.2.1 Simulation Localization

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 utilized previously for hair-hair and hair-object collision detection in Chapter 3.4.2, where sections of hair located in the same cell are tested against each other for overlap as are hair sections and polygons of objects in the scene. Here, this process is extended to all features of hair simulation, not just collision detection.

Insertion into the Grid

As explained in Chapter 3.4.2, 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 6.3(a) shows
the grid cells that contain hair geometry.

When 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 the 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 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. Finding and handling potential collisions for hair-hair and hair-object interactions are described in Chapter 3.4.2.

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 6.3. 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.

6.2.2 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 in Chapter 3.6, 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, discussed in Chapter 4. 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. Most styling 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 in Chapter 3.6, 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.

As the previous section explained, grid cells the user is focusing on are determined by inserting the cone or triangle of influence into the grid. These grid cells are then "activated" and each hair section in the cells become active as well. 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.

6.3 User Interaction and Applications

Given the LOD framework for hair simulation and rendering and the simulation localization technique described in the previous section, a user can now directly interact with hair through the 3D user interface and employ operations commonly performed in hair salons. The operations supported in this system include applying water, hairspray, and mousse to the hair, grabbing and moving sections of hair, using a hairdryer, and cutting the hair. The rest of this section describes each of these hair modeling features and their influences on hair dynamics.

6.3.1 Hair Cutting

Cutting hair is crucial for changing a hair's style. A common approach for specifying hair length has been to control the length of the hair through a parameter that either makes the hair grow or shorten. Some works have allowed the length parameter to change locally through either curve drawing [XY01, KN02], or via a 2D length map that correlates to the distribution of strands on the scalp [CK05]. Alternatively, Lee and Ko [LK01] presented a method to cut hair with a cutting surface; hairs that intersect the surface are clipped to the surface shape.

The cutting method I present builds on the work of Lee and Ko [LK01]. The method I introduce strives to emulate the hair cutting process in a real salon that uses scissors to trim the hair. All of the features of the cut are modeled, including simulating the hair that falls down after being cut. The location for cutting is defined by a triangle formed by the space between the open blades of scissors. Figure 6.4 illustrates the cutting process showing the blades of the virtual scissors and the hair skeleton to be cut.

In the simulation, hair is modeled using the dual-skeleton system described in Chapter
5.2. In this system, the local-skeleton defines the location of the hair geometry, thus it is used to determine intersections with the cutting triangle. In the rest of this section, I will refer to the dual-skeleton system simply as the "skeleton" for simplicity. Hair skeletons that intersect the triangle are then cut. At the cutting location, the skeleton $S$ is split into two separate skeletons, $S_1$ and $S_2$; $S_1$ remains attached to the scalp, while the latter, $S_2$, falls down.

![Diagram](image)

Figure 6.4: (a) Open blades of scissors define cutting triangle (shown in purple) (b) and (c) Grey skeleton (top) remains attached to scalp, red skeleton (bottom) falls down after cut.

At the point where the skeleton $S$ intersects the cutting triangle two new control points are created. One control point becomes the last point of skeleton $S_1$, while the second becomes the first point of $S_2$ that falls away after the cut, see Figure 6.4. The geometry of the fallen hairs remains consistent with the geometry of the hair below the sever point before the cut is performed; curliness, wetness, hair distribution and other properties are maintained in the fallen hair segments.

Once the hair is cut, there is less mass pulling on the hair at the roots. The mass is decreased along the skeleton to reflect this change. Skeleton $S_2$ is simulated with two free end-points; dynamics, including collision detection and response, are still enforced. The hair that is falling down is subject to large motion due to the force of gravity. Thus, it is always
simulated with high detail to capture the intricate effects.

Since the fallen hair is no longer attached to the avatar's scalp, hierarchy transitions are difficult as sibling hair groups can easily move far apart from each other as they fall making a transition into a coarser parent representation rare. As a result, simulating and rendering these fallen hair groups even after they have fallen to the floor and are no longer within the field of view of the camera could grow cumbersome to the overall system. To help alleviate this computation overhead, once these hair sections are outside the field of view of the camera and have fallen below a designated height (around the waist of the avatar) they are no longer simulated or rendered.

![Haircut Examples](image.png)

Figure 6.5: Example of hair cutting, far right shows final style.

### 6.3.2 Applying Water

Wet hair is modeled using the techniques described in Chapter 5. When water is applied to the hair, the mass points of the global-skeleton become heavier with the mass of the water. The overall motion of the hair is limited due to the extra weight and if the hair is curly, the global-skeleton stretches under the extra weight and the curls lengthen as expected. The volume of the hair in the areas where water is applied is decreased by constricting the radius of the current hair representation (strand grouping, cluster, or strip); these hair segments are then rendered to show the wet appearance as described by Chapter 5.4.
6.3.3 Applying Hairpsray and Mousse

Hairspray is simulated on the hair by increasing the spring stiffness of the global-skeleton where it is applied. Moreover, dynamic bonds (discussed in Chapter 5.3.3) are added between sections of hair that are in contact when the hairspray is applied. Dynamic bonds model the adhesive quality of hairspray to make the hair move as a group throughout subsequent motions.

Though the effects of mousse vary by brand, the system models a “volumizing” mousse, which adds volume to hair. Volume is injected into the hair model by growing the radii of the hair sections it affects using the flexible geometric structure explained in Chapter 5.3.2. This process makes the hair fuller without adding more strands or skeleton models.

6.3.4 Grabbing and Moving Hair

In real hair salons, stylists are frequently grabbing and moving sections of hair to desired locations. Typically when hair is clasped by a person, a group of hair is selected at a local point along the strands. To simulate this effect, the user presses the stylus button and the control points that fall within the cone of influence of the grab function are determined. Among these selected control points, only one control point per skeleton is permitted to be grabbed; this rule correlates to the localized grabbing of hair while styling. If multiple control points of a single dual-skeleton fall within the cone, the point that comes closest to the cone’s center is chosen.

At this point, control points are in a grabbed state and are referred to as grabbed-points. In the grabbed state, as the user moves the stylus, the grabbed-point will follow the motion of the stylus. A grabbed-point cannot be pulled beyond its normal reach span (decided by its position in the dual-skeleton). The length of the hair is always maintained so that the lengths above the grabbed-point and below it are of consistent lengths while the point is moving. The constant lengths of the local-skeletons are preserved.

When the user releases the grabbed-point(s), he or she releases the button of the stylus and the former grabbed-points will fall due to gravity. Figure 6.6 shows the user grabbing
Figure 6.6: (a) User grabs and pulls a section of hair (b) User releases grip and hair falls back to place.

and releasing a section of hair.

6.3.5 Hairdryer

Hairdryers are one of the most common tools in a hair salon. When the stylus button is pressed, a strong constant force is applied in the direction the user has oriented the stylus. Any control points that fall within the cone of influence receive this strong force. Moreover, if a wet control point (see Section 6.3.2) is influenced by the hairdryer, the control point will "dry"; the amount of water will decrease over the length of exposure dependent on the strength of the hairdryer force.

The hairdryer also has the ability to break the dynamic bonds created by hairspray (Section 6.3.3). When two control points linked by a dynamic bond are moved apart past a given threshold, the bond will break. The stiffened spring forces, however, remain strong and can only be dissolved by moisture (i.e. water application).
6.4 Hierarchy Inheritance

The applications described in the previous section change the properties of the hair whether permanently or temporarily; therefore a method for propagating these properties through LOD switching has to be employed. The effects of water, hairspray, mousse, and cutting are propagated up the hair hierarchy by averaging the values of the children dual-skeletons when they transform into their parent. Therefore, water values, spring stiffness, thickness, and length values are averaged. A parent skeleton also takes over the same dynamic bonds of its children; any hair sections connected to a child skeleton then becomes connected to its parent.

Traveling down the hierarchy, child skeletons inherit the same values as their parents. For example, a parent control point having 80% wetness will pass on 80% wetness to each of its children. The water value is not divided among the children because water would be lost throughout the system. The values are averaged when going up the hierarchy because the water is spread throughout all of the hairs. A hair section with 0% wetness combines with a section of 100% wetness and the water spreads throughout both resulting in 50% wetness for its parent.

Similarly, spring forces, thickness, and lengths are passed onto the children when traveling down the hierarchy. Any dynamic bonds that are created on the parent will be passed on to its children. Moreover, if hairspray has been applied to a parent control point, when it splits into its children dynamic bonds then connect each child.

Cutting hair has the most significant impact on the hair hierarchy because the number of control points per skeleton may be altered. Inheriting a cut through adaptive splitting is a simple process. If a hair group loses the last \( x \) number of controls points from a cut, during a transition, each child skeleton will lose the same \( x \) number of control points and all the child hair groups will be the same length as the parent. In the opposite direction, adaptive merging for simplifying the simulation is more constrained. As child skeletons prepare to transition into their parent, the number of control points on each child is compared. If all the child skeletons have been cut and still contain the same number of control points, then
their lengths are averaged and the transition into the parent occurs. However, if the child skeletons have been cut but do not contain the same number of control points, the transition is not permitted. Using the simulation localization technique described in Section 6.2, the dynamic simulation of the hair groups can still be accelerated if they lie in dormant cells.

6.5 Results and Discussion

The virtual hair salon system presented in this chapter demonstrates the usefulness of the level of detail framework for interactive hairstyling. While the initial hairstyles are loaded as a pre-process, the system allows for a user to dynamically alter the properties through several common hair salon applications.

6.5.1 Discussion

The methods introduced in this chapter allow for physically-based user interaction with dynamic hair while modeling several common hair salon applications. Figure 6.10 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.

Dynamic simulation, including implicit integration, LOD selection, hair applications (wetting, cutting, etc.), and collision detection, for a hair model shown in Figure 6.1 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 6.7 shows a performance comparison of the average runtimes between the LODs with simulation localization and a standard wisp implementation (consisting of 296 wisps) in the areas of Simulation (dynamics, collision detection, LOD selection, application execution), Rendering (shadow and lighting computations on the GPU) and Render Update (subdivision of curves and surfaces and passing render information to the GPU). It is obvious that the dynamic simulation component has been accelerated the most of all the areas. Figure 6.8 shows a more detailed performance comparison over the course of an entire simulation between LODs, LODs coupled with simulation localization, and wisps (which are used as the baseline of comparison). 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.

Finally, Figure 6.9 illustrates the breakdown of the simulation component of the LODs with simulation localization. The most time-consuming components in simulation are hair-hair and hair-object collision detection, as expected. Selecting an LOD adds only a small amount of extra time to the simulation (about 1% of the overall simulation time) while accelerating the overall simulation substantially as shown in Figures 6.7 and 6.8.

### 6.5.2 Limitations and Future Work

The system demonstrated in this chapter shows the usefulness of multi-resolution techniques for interactive hair modeling. The current property inheritance implementation focuses on maintaining the overall simulation state. It is possible to lose some specific details through transitions. For example, if a hair group with 0% wetness merges with a group of 100% the resulting parent will have 50% wetness. If this parent immediately splits back into its children, each child will have 50% wetness rather than their initial 0% and 100% values, respectively. However, a hairdryer could be used on the parent before it transitions back into its children, decreasing its wetness to 20%; it would then be incorrect for the children to inherit 0% and 100% because it would not reflect the drying process. One possibility to alleviate this loss of accuracy would be for each hair group to maintain a timestamp to track when its properties were changed and what the values were at the time. During a transition, the timestamps of
Figure 6.7: Performance comparison between hair wisps and LODs coupled with simulation localization. Rendering consists of shadow and lighting computations on the GPU. Render update includes subdivision of curves and surfaces as well as passing render information to the GPU. Simulation includes implicit integration for dynamics, LOD selection, collision detection (both hair-hair and hair-object), and application processing (blow-dryer, wetting, etc.). The simulation component is accelerated the most in the virtual hair salon, primarily due to the simulation localization feature.

the parent and the children could then be compared to see what the most recent values are.

Moreover, currently the cutting application only recognizes when the local-skeleton actually intersects with the cutting triangle. Situations where the hair geometry intersects the triangle but the skeleton does not are ignored. The cutting application could be made more accurate to test for intersections of the hairs SSVs used for collision with the cutting triangle. The resulting cuts could then duplicate finer details of the application, though the intersection test would be more complex.

There are several advancements that can also be performed as future work to either increase the types of styles that can be created or increase the user’s experience:

- Clasping hair to mimic the effects of barrettes or bobby pins to allow for hairstyles such as ponytails; the simulation could then be performed using constrained dynamics.

- Use of two-handed interaction with hair through haptic gloves. More interesting inter-
Figure 6.8: Simulation Performance Comparison. Shows the factor of speed-up for LODs with simulation 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 outperform both wisps and LODs alone, though the simulation varies over time as the user employs different applications.

actions can be modeled with this approach including the creation of braids and other more complex hairstyles. Furthermore, we would be able to model physically-based methods for curling hair that involve rollers or curling irons where the hair is held in one hand and the roller or curling iron in the other.

- Dynamic combing of the hair.

- Integrating high-fidelity force-feedback into the haptic system for more realistic user interaction.

- Explore the use of adaptive grids so the resolution of the grid can vary along with the simulation, providing another level of multi-resolution interaction. Currently, we have chosen a static grid resolution so each grid cell has the dimensions of the length of the average hair segment size (the distance between two local-skeleton control points),
Figure 6.9: Simulation Breakdown. Illustrates the percentage breakdown of each simulation factor using LODs with simulation localization. Note hair-hair and hair-object collision detection still make-up the majority of the simulation.

which we have empirically found to be an optimal grid resolution.

6.6 Summary

In this chapter I have presented an intuitive 3D user interface and methods for animating hair that allow for a user to interactively manipulate hair through several common hair salon applications. This system provides a level of user interaction that has before been too complex to achieve. The primary contributions presented in this chapter include:

- **3D User interface** that attains three-dimensional input from the user through a PHANToM stylus and allows for the user to directly manipulate dynamic hair. Moreover, the user is able to change tools dynamically using the stylus through the 2D menu icons located on the screen.

- **Simulation localization** technique that provides an additional level of simulation control of the hair by effectively turning-off the dynamic simulation of hair sections that have no activity or user interaction. This technique is particularly useful for hairstyle
creation due to the localized focus of a user when styling hair and is able to accelerate hair simulation so a user can interact with it.

- **Hair salon applications**: I discussed the creation of several common hair salon applications that the user can employ to manipulate the hair. Of particular interest is the hair cutting function, which is the first implementation that simulates the hair falling down as a result of the cut; this feature helps to add to the authenticity of the experience.

- **Hierarchy inheritance** methods that propagate the properties of the hair up or down the hair hierarchy. The hierarchy inheritance allows the user's changes to the hair to remain consistent from one level to the next.

The virtual hair salon system presented in this chapter combines the techniques presented in Chapters 3, 4 and 5 for the purpose of direct user manipulation of dynamic hair, an accomplishment that has never been achieved previously to the best of our knowledge. These results help to illustrate the benefits levels-of-detail offer to modeling hair.
Figure 6.10: Comparison between real (left) and virtual (right) 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.
Chapter 7

Conclusion

In this dissertation, I have presented techniques to dynamically change the hair model, allowing a simulation to balance between the visual fidelity and performance speed of the animated hair. The methods I have presented use level-of-detail techniques to control this balance. In this chapter, I summarize the main results presented in this dissertation and describe some limitations and possible future research directions of this work.

7.1 Summary of Results

I have presented methods for modeling hair using levels-of-detail that provide a balance between the computational performance and the visual fidelity of animated hair. The goal of my work was to determine on-the-fly which areas of the hair are the most significant to the simulation and to the viewer, and then to dynamically allocate the majority of computational resources towards modeling these areas. The remainder of the hair then deemed less important to the simulation is modeled with a coarser resolution, thereby accelerating the simulation and rendering of the hair while preserving the overall visual fidelity of the animated hair.

I described several methods to measure the significance of a hair section to the simulation. The criteria I identified for judging this significance included the hair's visibility, viewing distance, motion, and the user's interaction with the hair. The hair's visibility is measured by determining if the hair is outside the field of view of the camera or if the hair is occluded by some object in the scene. The viewing distance criterion relates to the amount of screen space
a section of hair covers. The greater the distance of the hair from the viewer, the less screen space the hair covers and, thus, the less observable the hair's detail is. I then account for the distance of the hair from the viewer by measuring the amount of screen space a section of hair covers and comparing this value to several thresholds to choose the appropriate resolution for the hair. The greater the motion of the hair, the finer the simulation needs to be to capture intricate details. The velocity of the hair is used as a measure of the hair's motion and is compared against thresholds for choosing the appropriate LOD resolution. Finally, the user's interaction with the hair constitutes the final criterion for measuring the hair's significance. As a user interacts with the hair, the physical properties, motion, and/or visual appearance of the hair can be altered. The regions of the hair with which the user interacts are thus simulated with fine detail to capture the large or minute alterations the user creates.

Together, these criteria and the methods for measuring them are the first presented, to the best of my knowledge, for calculating the significance of a hair group to a simulation. Furthermore, I presented techniques for dynamically changing the resolution of a hair group as simulation conditions change based on the aforementioned criteria. Using these techniques, hairs of low importance are automatically simulated and rendered with coarse detail, thereby accelerating the overall modeling of the hair without compromising the fidelity of the hairs deemed most important.

The level-of-detail framework I introduced is based on three discrete representations for modeling hair: hair strips, hair clusters and hair strands. Each representation provides a different level of computational speed and visual fidelity, as shown by performance and visual comparisons in Chapter 3. A strip models the largest portion of hairs, provides the fastest performance speed, but also has the lowest visual fidelity towards modeling hair in comparison with the other two representations. The clusters improve upon the visual faithfulness but lower the computational speed of the hair in comparison to the strips. Finally, the hair strands have the highest visual fidelity towards modeling hair but the slowest performance of the three representations.

The hair hierarchy, presented in Chapter 4, improves upon the use of the three discrete representations by providing a more natural progression in detail making the transitions
between LODs less obvious. Moreover, the hair hierarchy models dynamic clustering effects that occur as groups of hair merge and split. These dynamic clustering effects are difficult for many methods to model and increase the visual fidelity of the overall system. I also presented methods for adaptive splitting and merging of hair groups through the unified use of the base skeleton for each representation. As level-of-detail transitions take place, hair properties are passed from one level to the next through the base skeleton.

Additionally, I presented a number of techniques for accelerating certain essential hair modeling algorithms that can be used with or without the LOD framework. I derived an implicit integration scheme for handling hair dynamics that lead to larger time steps with greater stability. I also introduced a collision detection and response method based on the use of swept sphere volumes (SSVs) as bounding volumes for the hair. SSVs provide a tight fit around the geometry of each LOD hair representation as well as a simple collision detection test that can be used for both hair-hair and hair-body interactions. I showed the effectiveness of the collision detection scheme through several simulations, including the intricate case when groups of hair are twisted around each other and allowed to unravel.

Also, in this dissertation I discussed various hair rendering schemes that are implemented on graphics hardware for accelerated lighting and shadowing computations. Though these rendering algorithms have been designed for fast computation, they are still able to model many of the newly observed details of hair fibers, including the multiple modes of light scattering measured by Marschner et al. [MJC+03].

Moreover, I illustrated some ways water and styling products change numerous properties of hair. While little to no work has been provided previously for modeling such features, I presented several techniques for effectively capturing these influences on hair. Following the measurements obtained on real hair in biological and chemical literature, wet hair is simulated by increasing the mass by up to 45%, increasing the stiffness of the hair by a factor of three, and altering the geometric representation of the hair using radius (or width) contraction. Hair with styling products applied is modeled by adjusting the mass and spring stiffness (particularly increasing the length stiffness to maintain tight curls), altering the volume of the hair (which can be increased or decreased based on the type of product used), and
through dynamic bonds that model the adhesive nature of most fixative products. Moreover, rendering of the hair is altered as the hair becomes wet; wet hair appears darker and shinier in comparison to dry hair. These rendering effects are modeled by parameterizing the key values influenced by water in the light scattering and shadow computation algorithms.

I showed the effectiveness of these algorithms towards capturing the effects water and styling products have on hair by comparing the results with real wet hair and hair with hairspray applied. The image comparisons show that the primary changes on hair have been successfully simulated. These properties of hair can also be dynamically changed to allow for the application of water or styling products to the hair on-the-fly.

While it is difficult to meaningfully quantify the computational errors introduced by the use of simplified representations for modeling hair, the resulting simulations can be subjectively evaluated by performing comparisons on the visual quality of the simulated results. In this dissertation I illustrated side-by-side comparisons between hair simulated with the LOD framework and hair simulated with a constant number of strand groups, or wisps. While the visual quality of each simulation was comparable, the LOD framework was able to accelerate the dynamic simulation up to two orders of magnitude and the rendering by a factor of six, as shown through various performance comparisons.

To further illustrate the effectiveness of the level-of-detail framework for modeling hair, I have combined the techniques presented throughout this dissertation to create an interactive system for styling hair. The interactive virtual hair salon, discussed in Chapter 6, provides a user interface that obtains 3D input from the user for direct manipulation of dynamic hair. Moreover, I presented an additional acceleration technique that performs simulation localization based on the hair’s motion and the user’s interaction. The simulation localization is able to focus the computational power towards the areas of the hair with the highest activity. The interactive virtual hair salon uses the proposed level-of-detail techniques to allocate the majority of computational powers towards the hairs that are most significant towards the application, allowing the user to style dynamic hair. As a result, I have presented the first system, to the best of my knowledge, that allows a user to directly interact with dynamic hair.
7.2 Future Work

The results I have discussed in this dissertation illustrate the benefits level-of-detail techniques offer towards modeling hair. There are numerous potential research directions that can be extended from this work in the areas of hair simulation, rendering and styling. I have discussed a number of limitations of the current system throughout this dissertation and in this section I will summarize these limitations and suggest a few possible solutions. Moreover, I will discuss a number of improvements to the system that can potentially broaden the applicability of these methods.

7.2.1 Partial Visibility and Graphics Hardware

Currently the visibility of a hair group is decided by the hair being outside the field of view of the camera or by total occlusion by some object in the scene (such as the avatar). Meanwhile, a large portion of the hair volume can be occluded by other hairs. Due to the complexity of the hair volume, these hair self-occlusions can be difficult to ascertain. However, recent and continuing graphics hardware advances suggest that hair self-occlusions can be efficiently and accurately computed on the GPU using algorithms similar to computing hair self-shadowing effects.

The motion and viewing distance criteria used for hair LOD selection is based on comparisons against thresholds, where the thresholds are determined as a pre-process based on the user's requirements for visual fidelity versus computational performance of the animated hair. Hairs with root positions located at the base of the neck are less likely to be seen in detail by the viewer since they are most often covered by hairs located at higher positions on the scalp [LK01]. Following this observation, hairs positioned closer to the base of the neck are given higher threshold values for LOD transitions, making them more likely to stay at coarser resolution representations thereby accelerating the simulation.

While the observation that these hairs are unlikely to contribute much detail to the simulation is valid for most scenarios, it is possible for the avatar to turn upside down suddenly making these hairs far more prevalent to the simulation. As a result, the static choice for
threshold values could be limiting in various scenarios.

The current implementation is limited to check for full visibility of the hair sections. Partial occlusion is not considered in LOD transitions. It would be interesting to extend the current system to also consider partial visibility.

### 7.2.2 Hairstyle Creation

The virtual hair salon system presented in Chapter 6 incorporates the first proposed methods for allowing direct user interaction with dynamic hair. The results of this system are very encouraging for level-of-detail use in hairstyle creation although there are still many features that can be added to the system to broaden its applicability and to increase the users experience.

The 3D user input provided by the PHANToM stylus is useful for modeling a range of applications including cutting, wetting, blow-drying, etc. Incorporating two-handed interaction into the system through haptic gloves would offer an even wider array of applications to model. The user could grab and move hair in a manner more faithful to real-world styling allowing for intricate hairstyles to be modeled, such as through dynamic braiding.

Additionally, adding force feedback through either the stylus or two-handed interaction would also help to increase the faithfulness of the system. As the user grabs and pulls the hair, providing force feedback to the user would alert him or her that the hair cannot be pulled any farther, making the styling process seem more natural. Moreover, as the user grabs and moves the hair, collision detection between the hair and the head prevents the hair from penetrating the head, however there is currently no force feedback to alert him or her that a collision has taken place. Feeling the hair can help increase the realism of the system.

There are also more applications that can be added to the hairstyling system such as dynamic combing. Combing and brushing are common applications performed on hair and simulating this process can help to model additional hair features such as knots or other tangles in the hair. Moreover, incorporating force feedback with the combing could help to emphasize different properties of hair; for example, hair with hairspray in it will be more difficult to comb than hair without hairspray.
Another interesting hairstyling process to simulate is clamping hair similar to using a barrette or a bobby pin. Constrained dynamics can be incorporated into the simulation to adhere sections of hair at designated positions. Once again, many more interesting hairstyles can be created by modeling this effect.

The methods presented for modeling water and styling products on hair are still in the beginning stages. These techniques can be advanced by using exact measurements of different cosmetic products to create an accurate model for practical application in the cosmetics industry. Similarly, additional measurements of wet hair can be integrated for a precise model that would include the plasticity of wet hair.

7.2.3 Additional Limitations and Future Work

As with most LOD algorithms, aggressive switching can still lead to visual artifacts of the simulation and the amount of memory required is greater than modeling hair without LODs. It would be beneficial to investigate methods for alleviating these two factors. Additionally, the transitions between LODs occur completely from root to tip of the hair group. Incorporating a method similar to Bertails et al. [BKCN03] for splitting and merging groups starting at the tips of hair groups could extend the level-of-detail framework for more complex transitioning.

Moreover, in the current implementation, the creation of the hair hierarchy control structure is performed as a pre-process to the simulation. As more intricate hairstyles are created, such as braids and buns, the local neighborhoods of hair groups may change dynamically throughout the simulation desiring the information in the hair hierarchy to be updated on the fly. LOD merging and splitting based on hairstyle adjacency information can lead to more efficient dynamic simulation and rendering.

7.3 Conclusion

Hair modeling is an active and interesting area of research due to the intricacy of hair and the many different styles, properties and motions of hair. In this dissertation, I have
demonstrated the effectiveness of level-of-detail techniques for providing a balance between the visual quality and the computational performance of animated hair. I have also presented methods for interactively changing the properties of dynamic hair for hairstyling by accelerating the simulation and rendering of hair. Level-of-detail techniques provide many benefits towards modeling hair and, I believe, there are a number of additional exciting avenues to be explored in this area.
Bibliography


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


[Ney98] F. Neyret. Modeling animating and rendering complex scenes using volumetric
1998.


[NvdP98] I. Neulander and M. van de Panne. Rendering generalized cylinders with


simplification of particle system dynamics. In *Proc. of Computer Animation*, pages

[PB03] Deborah Patrick and Shaun Bangay. A lightwave 3d plug-in for modeling long hair
on virtual humans. In *Proceedings of the 2nd international conference on Computer
graphics, virtual Reality, visualisation and interaction in Africa*, pages 161–187. ACM

[PBL04] D. Patrick, S. Bangay, and A. Lobb. Modelling and rendering techniques for
african hairstyles. In *Proceedings of the 3rd international conference on Computer
graphics, virtual reality, visualisation and interaction in Africa*, pages 115–124. ACM

[PBS04] Sylvain Paris, Hector Briceño, and François Sillion. Capture of hair geometry from
multiple images. *ACM Transactions on Graphics (Proceedings of the SIGGRAPH


[PCP01] E. Plante, M-P. Cani, and P. Poulin. A layered wisp model for simulating
interactions inside long hair. In Nadia Magnenat-Thalmann Marie-Paule Cani,
Daniel Thalmann, editor, *Computer Animation and Simulation 2001Proceeding*,
workshop of Animation and Simulation.

accepted, June 2002.

[Per95] K. Perlin. Real time responsive animation with personality. *IEEE Transactions on

[PSO+05] G. Papagiannakis, S. Schertenleib, B. O’Kennedy, M. Arevalo-Poizat,
N. Magnenat-Thalmann, A. Stoddart, and D. Thalmann. Mixing virtual and real


