MODELING, RENDERING AND ANIMATING HUMAN HAIR

by

Tae-Yong Kim

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements of the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

December 2002

Doctoral Committee Form

To be replaced with the actual form

Acknowledgements

It was a long journey to complete this thesis. The journey was truly adventurous and challenging. Sometimes there were exciting findings along the way, while other times I was so depressed and it seemed that I would never finish this journey. I must confess that without my advisor, Ulrich Neumann, I could have never escaped out of it. Whenever frustrations came, he was there to encourage me. Whenever new findings came out, he was there to congratulate. He is also the best listener I have met in my life. He was always the first person to understand my problems and ideas no matter how significant or insignificant they were. Even when it seemed that I had no clues for some problems, he never forced me his ideas, but discussed all the possibilities and better formulated the problem in his words. Often the times, I would find solutions just by talking to him, and it was a truly amazing experience. I deeply thank him for not only being the source of encouragements, but also teaching me how to become a real scholar and how to do research.

I also would like to thank the other members of my thesis committee, Mathieu Desbrun and Jay Kuo. I am glad I could have them on my committee. Their suggestions and comments helped make this thesis complete and thorough. Through their scrutiny, I was able to evaluate and improve the thesis in totally different perspectives. Mathieu helped me understanding fundamental mathematical concepts, which can be priceless for my future research. I also would like to thank Ellis Horowitz and Ram Nevitia for serving as my qual committee. Ellis is an excellent teacher as well as a joyful racketball partner. Jay and Ram also helped me broadening my knowledge for graphics to images and vision in deeper level.

Special thanks go to J.P. Lewis for his countless help. He is a good friend, an excellent researcher, and a source of virtually any help. His comments for my papers and thesis turned invaluable. His admirable knowledge set always makes me wish I could become such a competent scholar like him some time. I really enjoyed the life as a member of Computer Graphics and

Immersive Technologies laboratory. As one of the early members of the lab, I have seen the lab growing into such a wonderful workplace. Dong Jin Cai, Ken Chen, Young Kwan Cho, Joe Demers, Jun Park, Ilmi Yoon were instrumental for me to find ways through the early stage of my study. Clint Chua, Reyes Enciso, Douglas Fidaleo, Junyong Noh, Bolan Jiang, and Jong Weon Lee have been excellent coworkers who helped me in some of the most critical stages. They are great individuals as well as researchers. Without their holistic supports, I could have never handled many critical problems. I also thank Anand Srinivasan for preparing all the lab equipments for me whenever necessary.

I deeply thank my family for their unconditional love and supports. Without them, none of this would have been possible. My parents always had faiths in me, even when I had disbelief in myself. Without their continuous sacrifice, I would have never done what I present here. I appreciate for their patience, love, and encouragement for their youngest son. My brother and sister always connected deep moral supports although they were thousands miles away at times. I would like to dedicate this thesis to my father who sacrificed everything for us. This thesis is not only my achievement, but also his.

Table of Contents

List of Tables

List of Figures

Abstract

As computer generated virtual humans become ubiquitous in many areas such as virtual agents, avatars in virtual worlds, and characters in games, these create strong demands for realistic human hair synthesis. Although methods for creating convincing face images and animations exist, it remains difficult to obtain the realistic appearance and behavior of human hair. As a consequence, hair is usually simplified or hidden with hats. However, hair is a very important cue for recognizing people. To faithfully model the natural appearance of a person, hair has to be accurately modeled, rendered, and animated.

The difficulties with human hair arise from several of its qualities. First, an average person's head has about 100,000 hair strands. With this number, even simplified models take vast amount of computing resources. Second, each hair strand's behavior is not trivial. Third, the interaction between the hairs should not be ignored. These interactions generate very important patterns that make one hairstyle distinguishable from others.

This thesis identifies the problems related to modeling human hair and presents related research efforts. Thin shell volume (TSV) method and Multiresolution hair modeling (MHM) method aim at modeling human hair due to hair-hair interaction. TSV method focuses on relatively flat hairstyles and structural details are added with virtual hair combing function on these hairstyles. MHM extends the idea of TSV method, enabling the user to model wide range of hairstyles at multiple level of detail. An interactive system is developed based on MHM method.

To render resulting strand-based hair models, a complete hair-rendering pipeline is developed. A novel shadow generation algorithm is presented that exploits existing graphics hardware. This approach is called opacity shadow maps and is much more efficient than previous hair shadow algorithms. A simple antialiasing algorithm using alpha blending is also presented. In conclusion, future extensions of the methods are proposed.

Chapter 1

1. Introduction

Computer generated virtual humans become ubiquitous in many areas such as films, virtual agents, avatars in virtual worlds, and characters in games. As the need to create realistic images of human faces increases, so does the need for human hair. Recent study reveals that hairstyle is the deciding factor on how a person appears on first impression [LaFrance 2001]. In a crowd scene, hairstyles are one of the most prominent features for a person. Cartoon drawings often use the same face prototype for different characters only varying hairstyles. Thus, hair is an indispensable cue for recognizing a human. Despite the importance of hair for humans, relatively little research efforts were made compared to the huge pool of researches on face modeling and animation. Due to many intrinsic difficulties associated with human hair, hairs are often simplified or even hidden (for example, with a hat). However, much realism is lost without hair. For example, a virtual avatar in a teleconferencing session will fail to provide the sense of immersion if the hairstyle does not match. Thus, realistic hair image generation will help any use of virtual characters, enhancing the realism of the virtual human.

However, hair remains a major obstacle in realistic human face synthesis. The volumetric nature of hair is simply not captured by surface models. Surface modeling methods can use high-density data acquired from range scanners. Such model acquisitions are not yet practical for hair since we lack suitable volumetric scanner technologies. Realistic hair image synthesis is difficult due to many reasons. First of all, there are too many hair strands to model effectively. Each hair strand is so pliable that it can be transformed to virtually any shape. Moreover, the interactions between hairs play important roles in generating various hairstyles. Due to hair-hair interaction, hairs tend to affect the shape of other hairs in a complicated manner. Simulating motion of hair becomes even more difficult due to these interactions. The discontinuity of hair and the thin geometry of individual hair strand necessitate specialized rendering methods to deal with such high frequency. Correct computation of lighting inside the hair volume is crucial to depict the underlying structure of any hairstyle, but still remains a challenging problem.

This chapter provides an overview on the problem of computer generated hair image synthesis. The difficulties in synthesizing faithfully looking human hair are described. A brief survey of related work is also provided. Finally, contribution of this thesis is summarized.

1.1 Problem Statement

Realizing computer generated human hair requires three related processes; modeling, rendering and animation. The difficulties with human hair arise in all of these processes. First of all, the sheer number of average human hair strands (100,000~150,000) complicates any task. With this number, even simplified simulation takes vast amount of computing resources. Moreover, the complicated structure and interaction between hairs present challenges in every aspect of graphics system.

In modeling, controlling the shape of every hair strand is an enormous task. A modeling method should provide the control over a large group of hair strands effectively, but still allow control over detailed variations. The structure (or style) of hairs as a group is often more meaningful than the properties of individual hair strands. The interactions between hairs (often referred to as hair-hair interaction) play an important role in generating these structures. Therefore, a modeling method should take into account the interactions as well as the individual properties.

Unlike other natural objects, the shape of human hair is routinely altered by humans. Through hairstyling, a hair volume can be transformed into virtually any imaginable shape. Ideally, a hair modeling method should be flexible enough to capture these complex styling procedures. Dealing with a large number of objects, the method needs to have a concise representation, to be used in computers with limited resources.

Due to the number of hair strands and the complicated shape of individual strand, rendering human hair often requires painful labors as well as demanding computational power. For example, more than 25% of rendering time for the movie 'Final Fantasy' was devoted solely to rendering human hairs. Also, virtual lights were manually positioned by a group of lighting artists to reproduce the complicated light propagation inside hair [Duncan 2001]. The followings are problems associated with hair rendering. Optically, hair has many interesting phenomena such as anistropic reflection, strong forward scattering, and translucency. A correct reflectance model is thus needed to compute reflectance off the individual hair geometry. While a local shading model approximates coloring of individual hair strand, interactions between hairs require efficient methods to compute such global illumination effects. In particular, the shadows between hairs provide essential visual cues for the structure of hair as a group. Hairs are inherently volumetric and the internal structures can be correctly illustrated only with proper shadows. Interestingly, a hair volume is often translucent, partly scattering and partly attenuating the lights from the surrounding environments. In some cases, multiple scattering between hair fibers tends to transfer incident lights on some positions to elsewhere, softening the shadow boundaries. Dealing with thin geometry such as hairs also requires special care for the undersampling problem when a point sample-based rendering algorithm is used (e.g. aliasing problem), to ensure the feel of smoothness.

Animating human hair is in itself a very challenging task. Each hair strand is very pliable and easily bent upon external forces such as gravity and collisions, while strongly resists linear stretches. This unique property often causes instability in numerical simulations. Thus, simulating the motion of a single hair strand is not a trivial problem [Pai 2002]. Hairs often collide with other parts of human body, especially the scalp and the shoulder. The hair motion is also affected by many subtle factors, such as wind forces, air drag, gravity, and head motion.

More interestingly, the static charge and friction between neighboring hairs often cause a group of hairs to form a cluster that moves coherently. Collisions between such clusters are important. It may appear unnatural if hairs interpenetrate each other. Especially in motion, if a cluster of hairs passes another cluster without any collision, it distracts the viewer and consequently hurt the realism of the animation. The clustering effects are not static. Due to hair-hair interaction, a large cluster tends to split into smaller ones (e.g. collision). On the other hand, a number of strands tend to form a cluster again under compression due to static charges. Due to the shape memory, hairs tend to retain its original shape (e.g. hairstyling). However, under excessive external forces, these memories are often lost. Thus, the preservation of the original styling brings another important issue. All of these problems are again aggravated with the number of hair strands to be processed.

Furthermore, the three components, modeling, rendering, and animation, are tightly related to each other. For example, without correct rendering, any hair model would be hard to perceive since it would lack in volumetric detail. Hair rendering is quite different from surface rendering in that while local shading is often enough for smooth surfaces, global illumination such as shadows is critical for hairs. Thus, although even a simple shading model (e.g. Phong shading) often suffices for displaying other parts of human body, the same analogy does not hold for hairs. Modeling and animation is also closely related to each other. Any modeling method that cannot be generalized to accommodate motion will be limited to a small class of static hairs that do not move. On the other hand, an easy-to-use, interactive animation tool can greatly help the modeling procedure since most hairstyles result from a series of hair-hair interactions or hair-object interaction. For example, hairs need to be stacked on top of each other in physically correct position, to give the sense of volume. An efficient, well-designed dynamics engine will greatly reduce the efforts of manually positioning each hair strand.

Due to the complexities and difficulties mentioned above, human hair is still an open problem in computer graphics and is considered one of the most challenging objects to realistically create in digital images. In summary, the ultimate requirements for any hair system include:

- Easy and flexible modeling capability so that the user can design any hairstyle. The user should be able to design a rough hairstyle quickly, but the system still has to allow fine-tuning to the strand level if necessary. The modeling system should not be limited to static pose of a single hairstyle, since hairs are deformable, moving objects.
- It should be able to produce realistic images of hair in a reasonable amount of time so that the user can quickly edit the hair model after visual feedback. It would be beneficial if the rendering system is scalable such that increasing the rendering time results in images of better quality in a user-controllable fashion.
- The dynamics engine should efficiently simulate the motion of individual hair strand as well as the interactions between hairs themselves and the interaction between hairs and other objects.

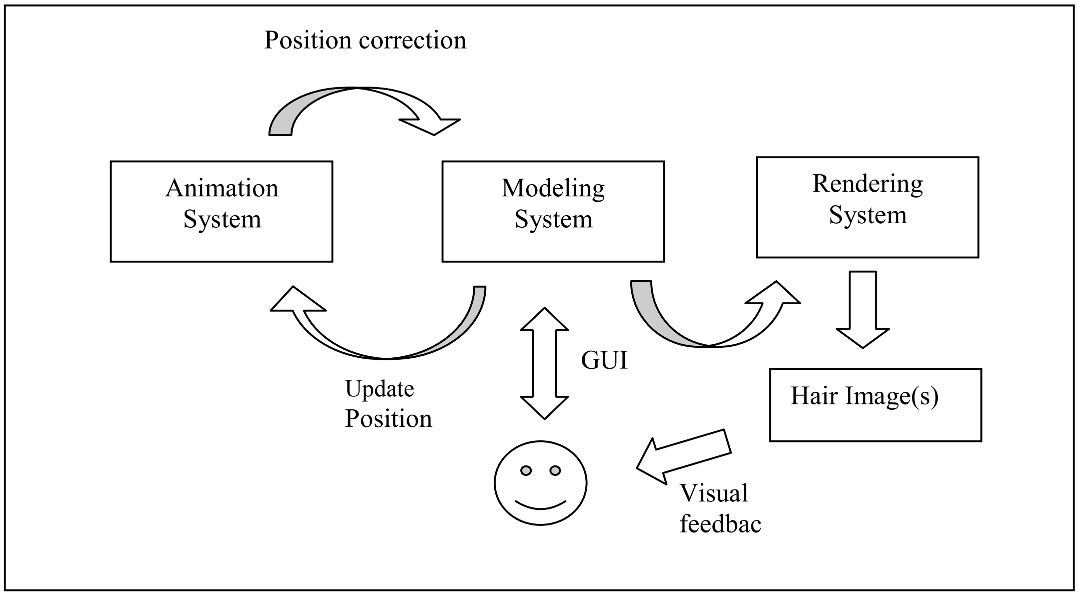


Figure 1-1. Schematic diagram of hair design process.

Interactive dynamics engine could be beneficial for the modeling process.

- The three components (modeling, rendering, and animation) should be integrated such that each component assists other components. For example, the animation system can help the user to correctly position hairs over the head. An efficient hair rendering system can provide the user with necessary visual feedback for refining the hair model or examining sequences of hair motion.

The relationship between the components is outlined in figure 1-1.

1.2 Related Work

In this section, previous research efforts are briefly reviewed. Interested readers should also refer to [Parke 1996] and [Thalmann 2000] for alternative surveys on hair research. The most natural representation is to represent each hair strand by a thin graphical primitive such as connected lines or a curve [Anjyo 1992] [Daldegan 1993] [Hadap 2001] [Lee 2001] [Rosenblum 1991]. As dealing with every individual hair strand is tedious and difficult, these methods often employ simplified physics models to simulate the dynamic behavior of hairs.

Alternatively, a group of hair strands are often replaced with a simpler representation. The surface representation approximates a group of hairs with a surface [Koh 2000]. The volumetric texture methods use a volume cell to represent a group of short hairs embedded in it [Kajiya 1989]. The cluster hair model approximates a group of strands with a generalized cylinder [Yang 2000].

1.2.1 Surface Representation

Polygonal surface was the first representation used for hair and is still one of the most popular methods. Csuri et al. pioneered to address the hair modeling problem [Csuri 1979]. They used a collection of polygons and associated reflection map to capture the complex appearance of hairs. As digitized hair texture became available through photograph and laser scanning, textured polygons were used to represent hair. The textures are often hand painted by the artists. The polygons serve as ‘canvases’ for the artist who ‘paints’ the hair texture on it. Alpha mapping, a texture map to specify the transparency of hair polygons, is used in [Koh 2000]. They extend the principle of surface representation with NURB surfaces. Due to the simplicity, alpha mapping combined with shadow textures are often the favorite choices among artists. In fact, this method generates relatively good images for static pose hair. However, this method is not based on the physical properties of hairs. As a result, it does not reproduce the true volumetric appearance of human hair. Hairs are volumetric and show complex view-dependent variation in geometry. Crude simplification using polygonal representation tends to make hairs look flat and skin-like in these systems. Also, detailed animation at strand level is not possible.

1.2.2 Static Pose Hair and Texels

Short hairs tend to stick to the surface and do not move much. Thus, these types of hairs are often assumed static. Perlin proposed hypertextures to create images of visually complex objects such as cloud, fluid, and hairs [Perlin 1989]. The noise based hair modeling was also implemented with real functions in [Sourin 1996]. These methods are limited to static pose of a hairstyle and local control of individual hair strands is difficult.

Miller modeled short animal fur using shaded wire frame attached to the surface [Miller 1988]. Each strand of hair is drawn with line strips using OpenGL. Kajiya and Kay [Kajiya 1989] introduced three-dimensional texture elements called *texels* to create a photorealistic, furry teddy bear image. The shading element is precalculated for a set of hairs and only the texels are used during rendering. This volumetric texture is extended at multiple scales by [Neyret 1995] [Neyret 1997]. The fakefur method also exploits the fact that hairs are very thin in normal viewing condition and thus can be approximated with simpler functions [Goldman 1997]. An appearance model for thin fur coat is developed based on a probabilistic lighting model. Gelder et al. presents an interactive animal fur modeling technique that produces a dense animal fur coat for polygonal models [Gelder 1997]. Bruderlin introduces a technique that takes into account clustering and breaking of short hair clusters to simulate the wetness of animal fur [Brudelin 1999].

For real-time rendering of animal fur, concentric textures are used to sample the volumetric texture functions, augmented with large-scale geometry [Lengyel 2000] [Lengyel 2001]. All the texture-based methods assume that some portions of complex geometry can be approximated and repeated over a surface. While this assumption holds for short hairs to some extent, it is hard to model long human hairs with this method since the repeatability assumption does not hold any more, as adjacent hair strands at the scalp split, curl, and move away from each other. Such styling effects require a significantly different modeling method from existing animal fur techniques [Lengyel 2001]. In general, human hairs are much longer than these methods can handle, and also animation is difficult with these methods.

1.2.4 Strand Based Model

In strand based hair model, every hair strand is represented with primitives such as cylinders, lines, and curves. With control over the appearance and motion of each primitive, these models are suited for long hair animation. Connected line segments are the most widely used representation due to their simplicity [Anjyo 1992] [Leblanc 1991] [Rosenblum 1991]. Each hair strand is represented as a connection of line segments. Typically the number of line segment runs around 10 ~ 80, depending on the desired level of model complexity and image resolution. For higher quality and smoother result, parametric curves such as bezier curves and spline curves, are also used as intermediate representations.

The trigonal prism is one of the early hair primitives [Watanabe 1992]. A trigonal prism is an approximation of a cylinder. The cross section of each trigonal prism is a triangle. Each trigonal prism has three sides and a hair strand is represented as connected prisms. Each hair strand is then rendered using the standard triangle rendering pipeline. In practice, line or curve representations are often favored in lieu of trigonal prisms since the trigonal prism generates a large number of small triangles subject to aliasing artifacts.

The strength of strand-based models mostly lies in animation. When affected by strong forces such as winds, hair strands move independently. Since the individual modeling of hair strand is extremely difficult, strand based models are often coupled with dynamics methods. For some class of hairstyles where the structures are relatively simple and hairs are mostly straight, strand based models can create quite convincing animations [Duncan 2001] [Hadap 2001]. However, more complex hairstyles are not easily modeled with these methods, even after repeated trial and error iterations of dynamics simulations.

1.2.5 Key Hair Approach

The most common approach in industry is the key hair approach, where a small number of characteristic hair strands are defined by the user and all the other strands are interpolated from this sparse set of key hairs. This is the main idea behind most commercial hair modeling packages [Berney 2000] [Fong 2001].

In the academic literatures, Watanabe and Suenaga [Watanabe 1992] and Daldegan et al. [Daldegan 1993] are perhaps the first to introduce the concept of using key hair, or a wisp. In their contexts, a wisp is a group of hair strands interpolated from three key hairs (one at each vertex of the triangle). Based on this wisp model, Chen et al. describes a system for interactively designing a hairstyle integrated with display of realistic face model [Chen 1999]. Recently, Chang et al. extends the key hair approach for animation where they simulate mutual hair-hair interactions between key hairs [Chang 2002].

Key hair (or guide hair) approach is the most powerful for short hair or relatively smooth hairstyles, where a small number of key hairs are sufficient to describe the overall structure of a hairstyle. However, as more complex hairstyles are desired, hair modeling becomes more tedious and time consuming. For some situations, designing just key hairs can consume 5 to 10 hours of modeling time [Thalmann 2000]. The common assumption that hair strands are parallel inside a wisp makes it hard to model rigorous strand variations such as curliness with a small number of random parameters to perturb wisps. Also, discontinuous clusters (e.g., due to combing) are not efficiently handled due to the underlying interpolation between key hair strands.

1.2.6 Discontinuous Cluster Model

An alternative way of hair grouping is to treat each group separately. A group of strands tend to form a coherent cluster due to static charges. It is sometimes convenient to model a hairstyle with a rough geometry such as generalized cylinders [Yang 2000] [Plante 2001] [Falk 2001] [Xu 2001], than controlling individual hair strand.

The cluster hair model represents each wisp as a generalized cylinder [Yang2000]. The method combines volumetric texture [Kajiya 1989] with wisp-based model. Instead of generating every single strand, the method represents hair strands inside a wisp as volume density function, which can be ray traced for improved rendering quality. An interactive GUI system based on the cluster model is introduced in [Xu 2001]. A similar modeling method was also used in the production of the movie 'shrek' [Falk 2001].

The benefit of the cluster model lies in the ease of control. Obviously, every hair strand cannot be manually modeled. In the cluster-based models, the user controls much smaller number of wisps. Although the cluster models provide a good tradeoff between controllability and level of realism, they often lack in generality in that the level of detail is limited to a single cluster level. Rigorous strand variations such as curliness are difficult to model, as noted by [Xu 2001].

Plante et al. [Plante 2001] introduce an animation method specifically designed for discontinuous wisps. Their simplified generalized cylinder model, a four-sided envelop of the hair wisp, quite effectively captures the deformable property of hair. The main focus of their work lies in simulating the interactions between these wisps in complex motion. Their hair grouping is fixed a priori and it is not shown how these methods can be generalized to more complex cases where a hair model consists of many small discontinuous wisps (such as curly hairs).

1.2.7 Particles, Fluid flow, and Vector field

One distinct property of hair, among others, is that a hair model is often as smooth as a continuous medium such as fluid, while as complex as thousands of discontinuous clusters. Fluid flow such as liquid or gas has similar properties and researchers have observed the similarity between fluid flow and hair. This aspect was exploited in the particle-based hair modeling method by Stam [Stam 1995], who considered each hair strand as the trajectory of a particle shot from the head. This way, a hair model can be modeled by repeatedly using a particle system. More involved use of fluid dynamics techniques appears in [Hadap 2000]. In this technique developed by Hadap and Thalmann, hair is considered as fluid flow. By placing components of fluid flow on the head, the user can easily create the global shape of a hairstyle, without worrying about the interactions between hairs. A similar, but different technique was developed by Yu, who instead used a user controllable vector fields to design the trajectory of hair growth [Yu 2001].

In reality, the shape of hair can be much more complex than these simple continuous medium models can handle. With these methods, detailed local variations are difficult to control as only a small set of arbitrary perturbation parameters are provided. Also, some man-made structures such as braids are not well handled since these structures become harder to describe with continuous fluid model or vector fields.

1.2.8 Hair Modeling from Photographs

One notable approach specific to hair modeling is to use photographs to automatically model hair. Kong and Nakajima are perhaps the first to address this issue [Kong1998a] [Kong 1998b]. In their methods, a few photographs are taken for a specific person and the boundary of the hair volume is constructed from the silhouettes extracted from these photographs. Hair strands are then randomly placed inside this volume with the guidance of a simple mass spring system to constrain each hair strand inside the volume. It is not shown how this method can be generalized to more complex styles, where volume extraction becomes harder due to holes and hair strand filling becomes non-trivial.

The recent work by Grabli et al. also attempts to automatically reconstruct a specific hair model from a set of photographs taken under a controlled lighting environment [Grabli 2002]. With known lighting condition and also known lighting model, tangential vectors of individual location on hair can be extracted. By linking these tangent vectors, they recover the geometry of each hair strand. Although promising, the method is yet prone to noise and thus does not recover a complete hair model. There still remain issues such as how to fill in the gaps between just a few hair strands recovered from the method, or how this method can be generalized to more complex cases where large portion of hair strands are not visible due to occlusion.

1.2.9 Hair Rendering

This subsection will briefly mention existing techniques for hair rendering (see chapter 5 and chapter 6 for more detailed discussions on hair rendering). In general, hair rendering methods are tightly coupled with the underlying model representations. Most of these rendering techniques share a local shading model, first developed by Kajiya and Kay [Kajiya 1989] and later modified by Banks [Banks 1994] and Goldman [Goldmann 1997].

In a normal viewing condition, the projected thickness of a hair strand is much smaller than the size of a pixel. Thus, a simple line drawing often causes severe aliasing effects. To create visually pleasing smooth images, pixel-blending buffers address the antialiased-rendering of small-scale hair strands [Leblanc 1991]. In this method, each hair strand is drawn as connected lines and the shaded color is blended into a pixel buffer. This method also proposes to use shadow buffer, a depth image from the light's point of view, to generate shadows between hairs and other objects. Since the shadow buffer stores discrete samples of depth information, the resulting hair image is subject to shadow aliasing (Figure 1-3b). Deep shadow maps addresses this issue and proposes to use a piecewise linear approximation of actual light transmittance rather than a single depth [Lokovic 2000].

Volumetric textures [Kajiya 1989] [Neyret 1995] [Neyret 1997] 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. Shadows can be calculated by traversing the volume density and by accumulating the opacity toward the light. The cost of ray-traversal is relatively low for short hairs, but can be high for long hairs. Also, in hair animation, such volume should be updated for every frame, making prefiltering inefficient.

The rendering method of the cluster hair model is similar to volumetric textures [Yang 2000]. 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, and thus hairs appear as if they always stay together.

1.2.10 Hair Animation

Strand based models are widely used for hair animation. Each hair strand is animated using one-dimensional angular differential equations [Anjyo 1992] [Lee 2001], mass spring simulation [Rosenblum 1991], and articulated body simulation [Chang 2002] [Hadap 2001]. See Chapter 7 for more detailed discussion on existing animation methods.

Cantilever beam approach simulates hair strands stacking over a head [Anjyo 1992]. After the hairs are positioned due to gravity, one dimensional angular dynamics method is used to simulate hair motions due to external forces such as wind. Lee and Ko propose multiple hull layers for more efficient handling of head-hair collision and hair-hair collisions [Lee 2001]. Although these methods are efficient in simulating individual hair strand's motion, the method has limitation in handling collisions. The collision between a hair strand and the head affects only lower part of the strand, and thus the method is limited in handling longer hairs that can touch shoulder.

Mass spring system provides an alternative to the angular dynamics method since collision handling is automatically propagated through spring connections. Each hair strand is represented with a set of masses connected with springs [Rosenblum 1991]. Hairs strongly resist linear stretches, resulting in a stiff system for mass-spring integration. Rosenblum et al. use an explicit Euler integration, known to be poor for such stiff system. As a result, the hair animation is limited to about 1,000 hair strands due to large computational requirement. The mass-spring configuration is also used to simulate the motion of a wisp [Plante 2001].

Early work on hair animation mostly ignores the interaction between hairs. The layered wisp model represents each hair wisp as a deformable viscous volume simulated with a mass spring system [Plante 2001]. Each hair wisp interacts with other wisps as well as the human body. An anistropic interaction model is used, where collisions are handled differently depending on relative directions between colliding wisps. Their work mainly deals with discontinuous wisp models. In contrast, Hadap and Thalmann consider dynamic hair a continum and fluid dynamics techniques are applied to the problem of hair-hair interaction [Hadap 2001]. While in motion, each hair strand exerts and receives a reaction force, calculated from the Lagrangian motion equation of hair medium. This method is particularly suited to smooth hairstyles, where clustering effects are ignorable. More recently, Chang et al. presented a method to simulate the mutual interactions between hairs, specifically in the context of the key hair methods [Chang 2002].

1.3 Thesis Overview

This thesis presents a series of my research efforts in hair modeling and rendering. Chapter 2 studies problems of hair image generation from various perspectives. The next two chapters deal with the hair modeling problem. Chapter 5 is dedicated to understanding the fundamental problems associated with hair rendering and chapter 6 presents practical solutions. The problem of animating hair is discussed in depth in chapter 7. Future work and ideas are also described in the chapter.

There are three main contributions made. The thin shell volume approach (Chapter 3, presented in IEEE computer animation 2000) builds on the idea that in hair modeling, structural details due to hair-to-hair interaction are as important as overall hair shapes. In this method, a rough hairstyle is defined with a set of parametric surfaces, while the distribution of hair strands are simulated through virtual hair combing functions. The idea is further extended in the multiresolution hair modeling system (Chapter 4, presented in ACM SIGGRAPH 2002). In this new system, a hair model is defined hierarchically, starting from a rough design to fine details down to strand level. A set of functions and tools are presented that ease otherwise tedious hair modeling process.

Chapter 5 and 6 deal with problems related to hair rendering. Chapter 5 discusses the fundamental theories and problems related to hair rendering. Ideas on improving existing hair shading methods are also presented. Chapter 6 presents an interactive framework for strand-based hair rendering. A novel shadow algorithm is introduced (presented in Eurographics rendering workshop 2001). This chapter also includes approximate visibility ordering algorithm for efficient antialiasing (the integrated interactive rendering algorithms were also partly presented in ACM SIGGRAPH 2002). The current issues and challenges in hair animation are summarized in chapter 7, along with ideas to extend the modeling technique for animation. This last chapter also presents ideas on extending current work in many aspects.

Chapter 2

2. Properties of Human Hair

Making digital images of human hair is a complicated process. Before going into any development, it can be useful to study various properties of human hair that affect the imaging process. This chapter discusses the issues related to the properties of human hair such as how individual hair reflect and transmit light, how hairs are perceived through human eyes, and how artists efficiently depict the complex structure of hairs. Understanding the imaging process will help in many aspects to build a practical image synthesis system of human hair. The key features of human hairs are analyzed based on visual perception model as well as physically based model. A simple image based hair modeling and animation system is then introduced in this context.

2.1 Properties of a Human Hair Fiber[1]

A human hair is, geometrically, a long, thin, curved cylinder. Depending on the individual, the hairs can vary in width from 0.03 mm to 0.09 mm. Hair has a strong specular component. Dark hair often appears shinier than lighter hair. This is because, in colored hair, part of the light is reflected at the hair fiber surface, and part enters the fiber and is scattered by reflecting off the interior. When light re-emerges, the diffuse component is increased. In dark hair, this diffuse component is absorbed before re-emerging, thus making the fiber appear shinier. A group of hairs reflect incoming lights strongly only perpendicularly to the direction of the cylinder. The strong

[1] The discussion in this section is excerpted from [Valkovic 1988]

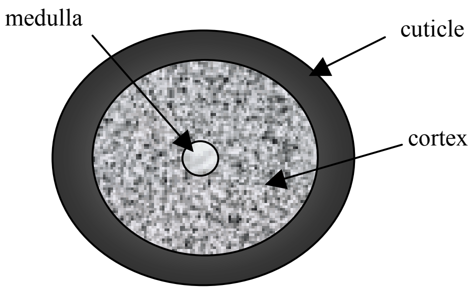


Figure 2-1: Cross section of a human hair fiber

highlights most noticeable in dark hairs are due to the anistropic (directional) nature of hair geometry.

Three layers decide the shape and color of each hair: *cuticle*, *cortex* and *medulla* (Figure 2-1). The *cuticle* is the outermost layer, which consists of hard, flat scales that overlap one another like roof shingles. This overlapping arrangement provides protection for the cortex and at the same time gives strength and flexibility to the hair. It is the *cuticle*'s flexibility that lets our hair be waved, curled, twisted, or braided without breaking. When hair is in good condition, the cuticle scales will lie flat, creating a smooth, reflective surface against which light can bounce. When hair is damaged and unhealthy, the cuticle scales break off and their edges become ragged. This uneven surface no longer mirrors light and causes hair to look dull, drab, and lifeless. The *cortex* is the layer immediately below the outer cuticle, making up about 75 to 90 percent of the hair's bulk. Pigment (hair color) is produced within this cortex and this important layer governs other hair properties such as direction of growth, size and diameter of hair strands, and texture and quality. The cross section of the cortex is different between straight, wavy, and curly hair. The *medulla* is the innermost layer of hair and its function is unknown. The medulla may even be completely absent – although its absence doesn't seem to affect the individual hair shaft in any way. The use of cosmetic products affects the property of hair in many ways, such as shape, the amount of bending, and color.

The condition of the cuticle is responsible for the outward appearance of the hair and dictates the attributes of feel and shine. In virgin hair, the cuticle platelets lay flat against each other and are firmly adhered to the cortex, giving rise to a very smooth feeling surface with a high degree of shine. If the hair is subjected to environmental or physical damage, the cuticle platelets can become chipped, raised or even detached completely, exposing the cortex underneath. This gives rise to hair in poor condition, which feels rough to the touch and is dull in appearance.

The most frequent cause of hair is mechanical abrasion, typically combing or brushing, often in combination with the use of hair treatment products such as perming lotions, colourants or even shampoos. The extent of physical damage depends, to a large extent, on the coefficient of friction between the comb or brush and the surface of the hair itself. This value rises dramatically if the hair is wet with the consequence that far more damage is normally inflicted to the hair if it is combed or brushed when wet, rather than when dry. The cuticle is also responsible for the water repellent properties of the hair. These properties are derived from the presence of lipids in the cuticle keratin which provides the hair shaft with a hydrophobic outer surface. If the cuticle is badly damaged or removed completely, water rapidly penetrates the cortex which subsequently swells, making the hair more prone to further mechanical damage.

In contrast to the cuticle the cortex, which forms the main body of the hair, is composed of a much softer, fibrous, crystalline keratin. The cortex itself is composed of a large number of cortical cells, held together by an intercellular binding material known as the cell membrane complex. The cortical cells themselves are long and thin in shape and, for this reason, are often referred to as spindle cells. The cortex is responsible for the mechanical properties of the hair and dictates the deformation and tensile behaviour of hair when it is subjected to external forces.

At any one time, there are typically over 100,000 hairs on the average healthy human scalp, each of these being associated with a hair follicle. The hair follicle is an extension of the epidermis, the outermost layer of the skin.

2.2 Perception Based Analysis of Hair Photographs

As one of the important goals of this thesis is realistic rendering of hairs, it can be helpful to start with a question, "What makes hair look like hair?" People often perceive hairs as a collection. One hairstyle is distinguished from another not by examining the individual hair strands separately, but by the overall structure and relationship between hair strands.

The attraction force resulting from static charge makes neighboring hair strands stick together. The cluster of hair strands exhibits a strong coherency in direction, color, and motion. At moderate distance, the clustered hair strands bear good resemblance to a hair strand. The clustering patterns are affected by many factors, such as the use of cosmetic products, combing, external forces, and collisions.

The structure of a group of hair strands is properly seen only through shadows between hairs. Unlike many smooth objects, shadows are crucial visible cues to tell whether a hair strand is behind or in front of other strands, thus presenting the sense of depth. Other depth cues such as perspective projection are not significant since the variation in projected thickness is very small. Shadows play an important role in coloring every hair strand differently even when all the strands have more or less the same amount of pigment. The shadow patterns vary over entire hairstyle and the hair images exhibit an interesting property – the patterns are often repeated at multiple scales.

Human hairs often have features that can be recognized at multiple scales (Figure 2-2). From distance, only a few clusters are clearly recognizable. In this scale, a cluster of possibly hundreds or thousands of hair strands much resembles a single hair strand, due to the coherence. But at closer view, this cluster in fact consists of many smaller ones. At the smallest scale, some fine features such as wavy individual hair strands become visible.

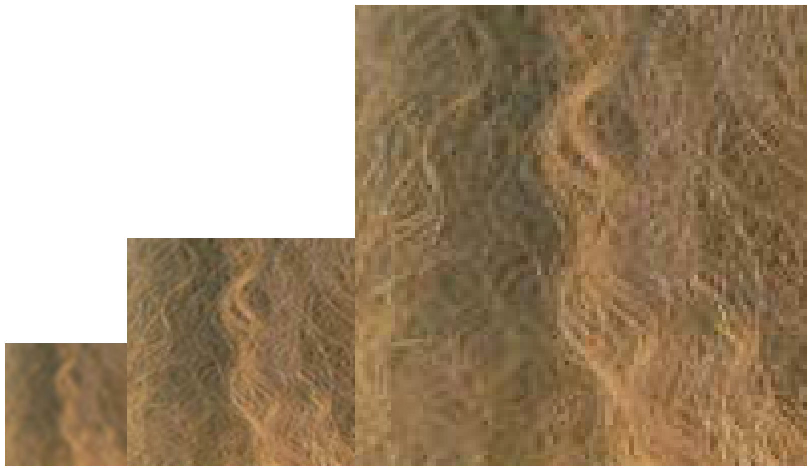


Figure 2-2. Photographs of human hairs at multiple scales.

In a signal processing term, a hair image exhibits strong high frequency energy as well as low frequency energy. Note that the hair region shows a strong energy on the high frequency image. This observation suggests that hair should be modeled at multiple scales, to account for all the detailed variations in hair geometry.

2.3 Artistic Drawings of Human Hair

Artists have long used appropriate level of abstraction when they draw a complex object such as hair. Instead of drawing every hair strand, artists often draw hairs only as perceived. Not all the visible cues are always used, but the drawings often manifest sufficient information on certain hairstyles. For example, illustrations often use anisotropy and self-shadowing as a tool to depict hairs. In this case, hairs are drawn mostly on cluster-basis, not strand-basis. The clustering is even more exaggerated in a painterly style. In many paintings, no single hair strand is drawn. To the painter's eye, hairs are perceived as 'clumps' that smoothly change the shape, color, and shadows. Some of important lighting effects such as translucency are captured with this subtle, yet effective painterly touch. The overall smooth feel of hairs is often the most important feature of some hairstyles. In this case, hair is perceived as fluid-like medium with fuzzy boundaries rather than a solid object with definite shape. The artists often omit details on purpose to make more succinct illustration, especially in hair drawings. It is not only because the hair drawing is tedious at strand level, but also because human eyes cannot differentiate such distressful high frequency details. In some cases, only silhouettes between the clusters are shaded.

2.4 Human Visual System

Human eyes are known to have different sensitivity over different frequency channel. The contrast sensitivity function (CSF) is a widely used measure for such sensitivity. The function measures the observed sensitivity to pre-determined contrast gratings.

2.4.1 Contrast Sensitivity Function

Many perception studies examine the perceptibility of contrast gratings, sinusoidal patterns that alternate between two extreme luminance values L_{max} and L_{min}. Contrast grating studies use

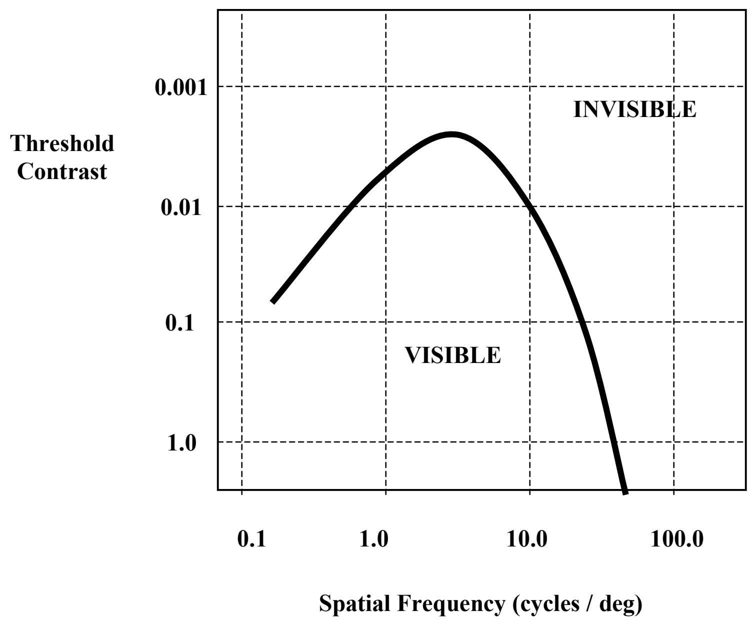


Figure 2-3: The contrast sensitivity function. Adapted from [Luebke 2001]

Michaelson contrast, defined as (L_{max} - L_{min}) / (L_{max} + L_{min}), and spatial frequency, defined as the number of cycles per degree of visual arc. The threshold contrast at a given spatial frequency is the minimum contrast that can be perceived in a grating of that frequency, and contrast sensitivity is defined as the reciprocal of threshold contrast. The contrast sensitivity function (CSF) plots contrast sensitivity against spatial frequency, and so describes the range of perceptible contrast gratings (Figure 2-3). The sensitivity is the largest around 1~10 cycles per degree, which is equivalent to looking at 10 point font in normal viewing condition (30cm away from the head). In this viewing condition, a hair strand takes about 0.10 ~ 0.15 degree, or 4~5 cycles per degree. It is interesting to note that the thickness of a hair strand in normal viewing condition coincides with the highest sensitivity level. This may explain why even a single hair strand can be still recognized under some lighting condition despite its thickness.

Indeed, this contradicts the observation by artists that every hair stand needs not be drawn. However, the sensitivity measure in CSF is based on contrast gratings. For hair strands to generate any contrast in images, they should be evenly spaced such that the distances between them are similar to the thickness. In a normal hairstyle, the distributions of hairs are more complicated than the gratings. Also the clustering of hairs result in less contrast since the shadows between hairs are not clearly seen. In reality, the majority of hairs are seen as a group, while a small portion of hair strands are clearly visible at strand level.

2.4.2 Effect of Visual Masking

Colorization of hair can be considered as the convolution of two signals; one from the local reflectance off the hair geometry, and the other from global scattering and occlusion of light between hairs (e.g. shadows). It is known that human eyes do not easily separate one signal from the mixed signal when both signals consist of high frequency channels [Bolin 1995]. In smooth hairstyles, both local variation in color and shadows are smooth, resulting in two low frequency signals. In more complicated hairstyles such as curly hairs, both signals are of high frequency. In this case, shadows are hard to differentiate from local color variation and the opposite is also true. This masking effect indicates in hair image synthesis that low frequency components are more important in a complex style. In other words, the cluster-level accuracy may be more meaningful than the strand level accuracy.

2.5 Image Based Hair Rendering and Animation[*]

This section briefly demonstrates a simple experimental hair animation system that exploits the discussions in the previous sections. The system is very simple, yet captures many important properties of human hair. Given a single image of human hair, the user interactively positions a hair wisp over the picture. The wisps are drawn with textures from the original photograph and then can be animated independently.

A layered model is used to represent a hair wisp. The layered model consists of the boundary of wisp defined with two space curves and the links connecting the control points of the curves . Inside this wisp, a number space curves are drawn that are evenly spaced in the parametric space of the wisp. Each space curve represents a hair strand or a small cluster of hair strands that looks similar to a single strand, which is drawn as a Catmull-Rom Interpolation curve. The control points of each space curve are linearly interpolated from the two boundary curves. The position of each control point also defines a texture coordinate from the input image.

In animation sessions, the user can interactively reposition the wisps by either directly moving the control points of boundary curves or by applying indirect forces. In a subsequent animation session, each curve is drawn using the original image as a texture. Each curve is drawn with a thickness of two pixels, to avoid aliasing. A simple mass spring system is used to animate the 2D wisp. The control points of the two boundary curves are linked with weak springs while the internal links between control points are connected with stronger springs. The mass spring configuration allows the wisps to stretch and compress while in motion.

[*] This work was done as my initial experiment to analyze the properties of hair in the summer of 1999.

2.6 Conclusion

In this chapter, the properties of human hairs in the context of digital image synthesis were discussed. The simple system presented in this chapter promised that realistic hair modeling may not necessarily involve an accurate modeling of every hair strand. Instead, a hair modeling system needs a representation that reproduces the structure and distribution of hair strands at proper level of detail. Given this motivation, the problem of modeling the structural detail of hair is discussed in the following two chapters.

It is a common practice in computer graphics to separate details from the overall geometry (e.g. texture maps, displacement maps). The same principle can be used in hair modeling. While the overall shape of a hairstyle is roughly represented with one method, details can be represented separately. The thin shell volume method discussed in the next chapter represents the shape with parametric surfaces while the details are added using virtual combing functions. The idea of separating details from the overall shape is pushed further in the multiresolution hair modeling technique (chapter 4), where hair wisps are modeled with generalized cylinders that are hierarchically defined.

Chapter 3

3. Thin Shell Volume Modeling[*]

The majority of human hair modeling methods can be classified as *geometry-based* [Watanabe 1991], [Anjyo1991]. These usually involve a large number of hair-strand primitives such as cylinders, surfaces, lines, and curves. With control over the appearance and motions of each primitive, these models are suited for long hair and animation. Most geometry-based methods model and render each hair strand independently. In reality, hairs are not perceived independently nor do they behave independently. Neighboring hairs tend to attract or repel each other. The resulting hair pattern provides an important 'signature' of hair appearance. These behaviors are very difficult to model, even for limited cases, thus *hair-hair interaction* (HHI) has been largely ignored in prior works.

This chapter introduces thin shell volume (TSV) method for hair modeling. In the method, the range of feasible hairstyles is limited to a set of long hairstyles that can be approximated with surfaces. The goal is to add more realistic appearance and behavior to this common set of human hairstyles. The overall shape of hairstyle is represented with parametric surfaces (coons patch), and the structure of individual hair strands are defined by virtual hair combing functions. The TSV method focuses mainly on the structural aspects of hair modeling, where the interaction between

[*] Presented in IEEE Computer Animation Conference, 2000.

hairs plays an important role. Virtually, any hair model based on surfaces such as NURBS or polygons can benefit from the method.

3.1 Shape Function

At large scales, long hairs tend to form smooth surfaces that can be approximated by polygons or parametric surfaces such as NURBS. At smaller scales, each hair strand is essentially a long, thin cylinder that can be represented by a curve. To correctly model long hair, it is necessary to take both scales into account. The TSV model builds on the simplicity of the surface representation while adding the fine appearance details of long hair. A thin shell volume is constructed on each surface. Inside the volume, hairs are associated with particles that move under constraints that model the hair-hair interaction during combing. Individual hair strands are represented by a set of particle positions inside the volume.

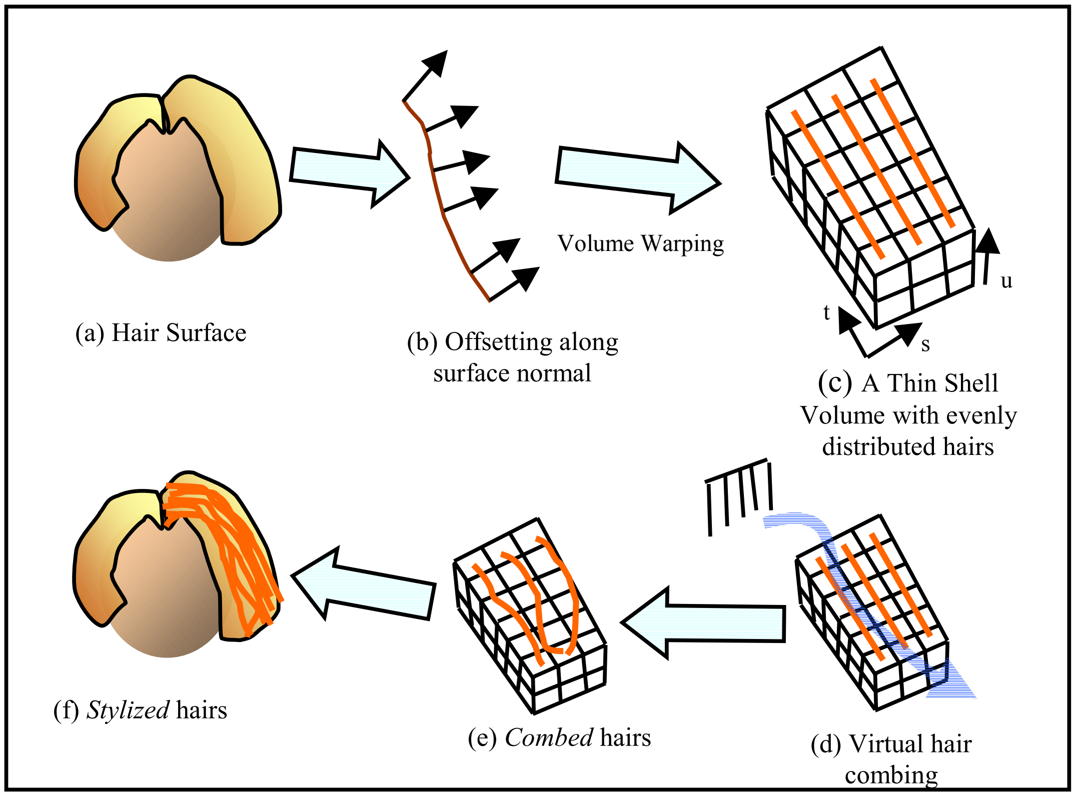


Figure 3-1 Overview of hair combing using a thin shell volume model.

Below is listed the procedures to perform virtual hair combing using the thin shell volume model (Figure 3-1). (a) First, a hairstyle is modeled with a set of parametric surfaces. (b) Each surface is added a *thickness* by offsetting the surface along its normal direction, generating a thin curved volume. (c) The curved volume is warped to a rectilinear 3D grid. Inside the gird, a number of hair strands are evenly distributed. Each cell of the grid is associated with a list of particles that reside inside it. (d) A virtual hair combing function is performed on the normalized volume. (e) The hair combing function redistributes hair particles, creating variations based on a HHI model, producing a set of *combed* hair particles. (f) Finally, each hair strand is reconstructed by connecting the corresponding hair particles. These *stylized* hairs are a group of curves where each curve represents one hair strand. The curves are passed to the rendering pipeline.

3.1.1 Notation and Terminology

Let us define some notations and terminology for use throughout this chapter. As shown in Figure 3-1(c), a TSV is parameterized by three variables ($\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}$). The $\vec{t}$ vector is aligned with the major direction of the hair flow. The $\vec{u}$ vector is aligned with the *depth* direction of the hair volume. The $\vec{s}$ vector is orthogonal to both $\vec{u}$ and $\vec{t}$. For simplicity, $\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}$ are normalized to range from 0.0 to 1.0. Along the $\vec{u}$ (depth) direction, the value of $\boldsymbol{u} = 0$ is assigned for the innermost hairs, and $\boldsymbol{u} = 1$ for the outermost hairs.

Let ($\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$) denote the integer grid index corresponding to a given ($\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}$). ($0<\boldsymbol{i}\leq L$, $0<\boldsymbol{j}\leq M$, $0< \boldsymbol{k}\leq N$, where L, M, N represents the grid resolution of the volume.) $\boldsymbol{V_{i,j,k}}$ identifies the ($\boldsymbol{i,j,k}$)th volumetric cell position. Particles $\boldsymbol{p}$ are distributed among the cells (Figure 3-2) representing each strand passing through a cell. Each cell maintains a list of particles residing in it. The density of particles defines the hair opacity $\boldsymbol{\rho}$ later used for shadowing. For a given particle, the cell containing it is determined by quantizing its $\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}$ position values.

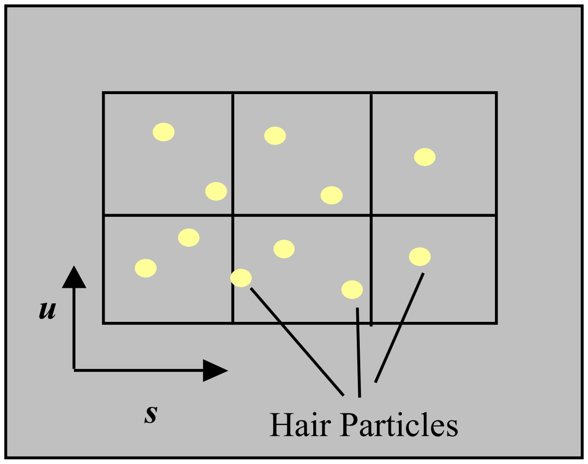


Figure 3-2. Cross section of a TSV.

3.1.2 TSV Generation and Volume Warping

The initial TSV shape is determined from the hair surface (or reference surface). We start from a reference surface (e.g., NURBS) parameterized as $\boldsymbol{P = S(s,t)}$.[2] The TSV is constructed by offsetting the surface along the normal (depth) direction, forming a thin curved volume (Figure 3-3).[3]

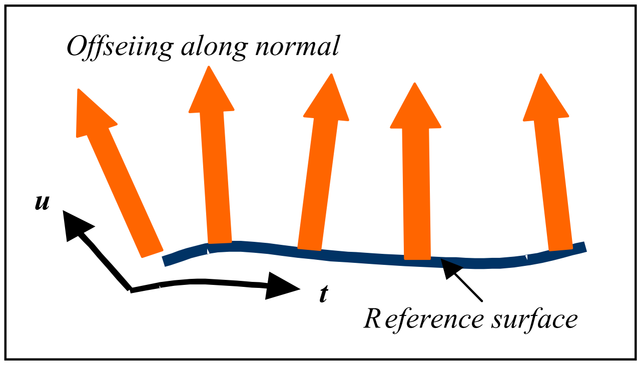


Figure 3-3. Generation of a thin shell volume

[2] In this work, Coons Patch is used as reference surface representation (refer to the Appendix 3-A.) Other surface representations are equally applicable as long as they can be represented in the form of $\boldsymbol{P} = \boldsymbol{S}(\boldsymbol{s},\boldsymbol{t})$

[3] Note that high curvatures and folded surfaces can cause self-intersections, but many useful reference surfaces are relatively smooth and free of these problems.

The reference surface is generally curved so the normal varies over the surface. The parameterization of a point ***p*** inside the thin shell volume is (s_p,t_p,u_p). The normal ***n(p)*** can be defined by taking the cross product of the partial derivatives of the reference surface ***S*** along $\vec{s}$ and $\vec{t}$.

$$n(p) = n(s_p, t_p) = \frac{\partial S(s_p, t_p)}{\partial s} \times \frac{\partial S(s_p, t_p)}{\partial t} \tag{3-1}$$

The world coordinate of ***p*** is then reconstructed by

$$p = S(s_p, t_p) + T \cdot u_p \cdot n(s_p, t_p) \tag{3-2}$$

where *T* is a scalar defining the thickness of the volume.

It is more convenient to compute the hair-hair interaction inside a regular rectilinear volume rather than in a curved volume, so we use an implicit volume warp. Any point ***p*** in the TSV with ***(s_p, t_p, u_p)*** can directly map to a corresponding point in a regular rectilinear volume we call a *normalized TSV*. The normalized volume has an orthonormal basis defined by the $\vec{s}, \vec{t}, \vec{u}$ vectors and greatly simplifies the description and calculation of our simulation for hair-hair interaction. In the remainder of this chapter, TSV is assumed normalized (the distortions caused by this simplification are not visually significant since they only affect the hair-hair interaction behavior.)

3.2 Virtual Hair Combing

The thin shell volume is a modeling method for distributing long hairs. The distribution of long hairs is usually very simplified for mathematical simplicity. Often simple random variations such as translation and rotation are used. These random variations do not, however, model hair-hair interaction and result in unrealistic hair distributions and appearances. Virtual hair combing provides a way of generating the hair patterns that often occur in real-world hair styling.

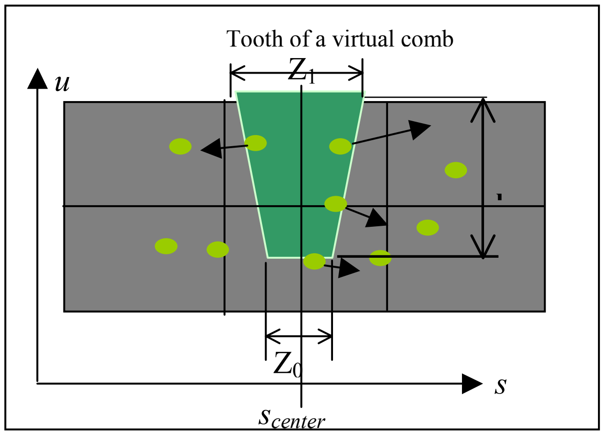


Figure 3-4. Virtual hair combing

A number of hair strands are generated inside the volume. They are uniformly distributed initially. For a TSV with a resolution of *L*x*M*x*N*, each hair is associated with *M* particles that lie along its path. There are roughly *n*/(*L*x*N*) particles in each cell. Each hair lies along the $\vec{t}$ direction. Each hair strand starts where ***t*** is close to zero, and ends where ***t*** is close to 1.

Virtual hair combing redistributes hair particles inside the TSV by simulating the effects of combing. Each hair that lies under the path of the comb moves under influence of the comb teeth. We generate various hair patterns by changing tooth shape, tooth distribution and the path of the comb. The cross-section of a comb tooth in the ***su*** plane is a trapezoidal radial influence function (Figure 3-4).

Each tooth *path* is modeled as a parametric curve on the ***st*** plane over the TSV. Each tooth of the comb has an influence function such that:

- Only the particles hit by the tooth are affected
- Each particle moves away from the center of the tooth's influence
- Each affected particle tries to follow the path of the comb
- The vertical motions of particles are generated in a way that combed hair moves upward (outward) to lie on top of the unaffected hair.

3.2.1 Particle Selection (pickup)

Each particle is tested if it is affected by the comb's tooth. We model the path of the tooth's center as a parametric curve ***C*** on the ***st*** plane, where $\boldsymbol{s_{center}}$ is the ***s***-coordinate of the tooth trapezoid center ($\boldsymbol{s_{center} = C(t)}$). We define an influence zone ***Z(u)*** along the ***u*** (depth) direction. ***Z(u)*** determines the vertical shape of the tooth. Our implementation models it as a trapezoid.

$$Z(u) = \begin{cases} \frac{1}{2}\{(1-u')Z_0 + u'Z_1\}, u' >= 0 \\ \varepsilon, otherwise \end{cases} \qquad (3\text{-}3)$$

$$u' = 1.0 + \frac{u - 1.0}{l}$$

In (3-3), ε denotes a very small value to avoid a division by zero in a later equation and *l* denotes the depth of the tooth. Using the distance from the center of the trapezoid, we define the zone influence function **Z*(s,u)*** as follows.

$$Z(s,u) = sign\ (R(s)) \cdot \cos(\frac{\pi}{2} \cdot clamp\ (\frac{R(s)}{Z(u)}))$$

$$R(s) = s - s_{center}$$

$$clamp\ (x) = \begin{cases} x, & \text{if } |x| \le 1 \\ 0, & \text{otherwise} \end{cases}, \quad sign\ (x) = \begin{cases} 1, & \text{if } x \ge 0 \\ -1, & \text{if } x < 0 \end{cases}$$

Z(s,u) is used as a probability function to check if a particle is inside the tooth's influence zone. Given a particle's position (***s,t,u***), we first select the nearest trazezoid based on ***t***, then we calculate ***Z(s,u)***. For each particle, we make a probabilistic decision whether or not the tooth picks up the particle. A random value is compared to the absolute value of ***Z(s,u)*** to determine the binary decision outcome. The sign of ***Z(s,u)*** determines the direction of the particle motion. A positive value moves the particle in the increasing-***s*** direction and a negative influence value moves it in the decreasing-***s*** direction.

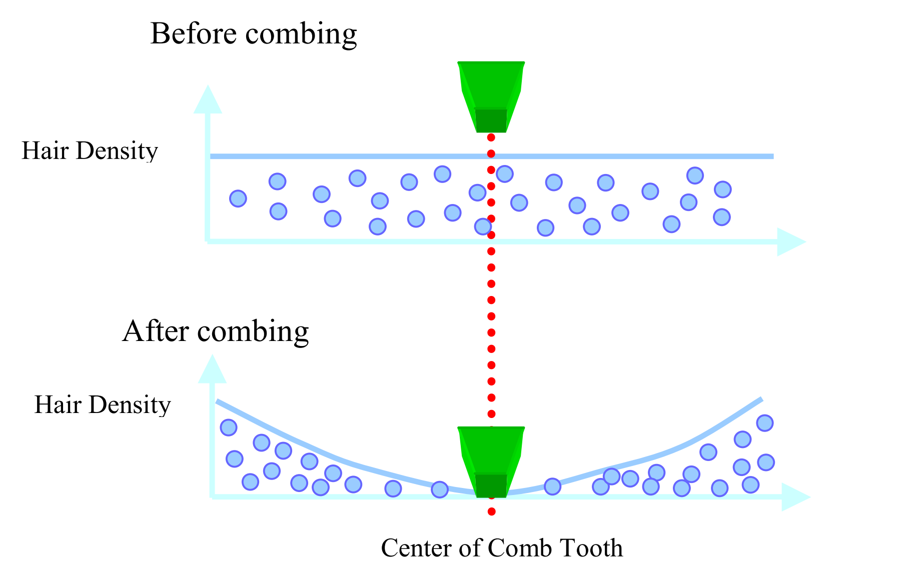


Figure 3-5 Hair density changes during combing

3.2.2 Particle Redistribution (s-direction)

Once a particle is picked up, we reposition the particle randomly (in ***s***) within the range of the tooth's influence zone. Due to the pickup function Z(s,u), the particles near the center of the influence zone are more likely to be repositioned. The particle distribution gets sparser at the center of the tooth's influence zone since hairs are pushed aside by the comb tooth. Figure 3-5 illustrates the particle density change due to combing.

3.2.3 Sweeping Effect

Random redistribution of the particles representing a hair strand may generate unwanted curliness. A hair strand should be bent accordingly to the path of the comb. To generate the *sweeping effect*, we use a simple scheme (Figure 3-6). For each hair strand, we find the first particle p_j influenced by the comb. Let the particle's parameterization be (s_j, t_j, u_j). We calculate the new ***s***-coordinate s_j' for this particle. We calculate the distance $r = | s_j' - s_{center}|$. For the

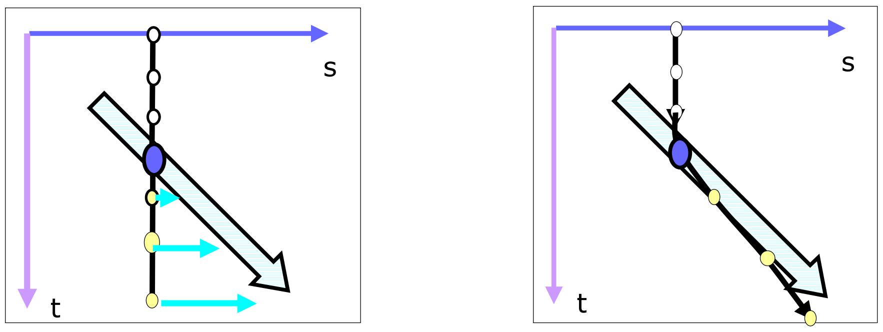


Figure 3-6: A hair strand is bent along the path of the comb's tooth.

following particles $\boldsymbol{p}_l$ ($l > j$), we add the distance to the path of the comb. That is, for a particle $\boldsymbol{p}_l$ parameterized by ($\boldsymbol{s}_l$, $\boldsymbol{t}_l$, $\boldsymbol{u}_l$), the new $\boldsymbol{s}$-coordinate $\boldsymbol{s}_l' = \boldsymbol{C}(\boldsymbol{t}_l) + \boldsymbol{r}$.

3.2.4 Vertical Motion (u-direction)

In a real combing, the combed hairs tend to be stacked on top of the remaining hairs. We simulate this behavior by moving combed particles upward (in $\boldsymbol{u}$-direction). Simply moving all the particles upward may generate a crowding problem after repeated iterations. Some (not combed) particles should be pushed down too. To simulate these effects, we swap the combed particle with a particle from the upper cell (Figure 3-7). If there is no upper cell or there is no particle in the upper cell, we move the particle upward with small amount (usually the size of cell in the $\boldsymbol{u}$-direction). To ensure that the particles lie inside the volume, any out-of-bound particle is reflected back to the inside of the volume. The virtual combing on a TSV can be performed repeatedly, varying the tooth distribution, tooth shape (in our case, Z_0, Z_1 and l), and the path of the comb.

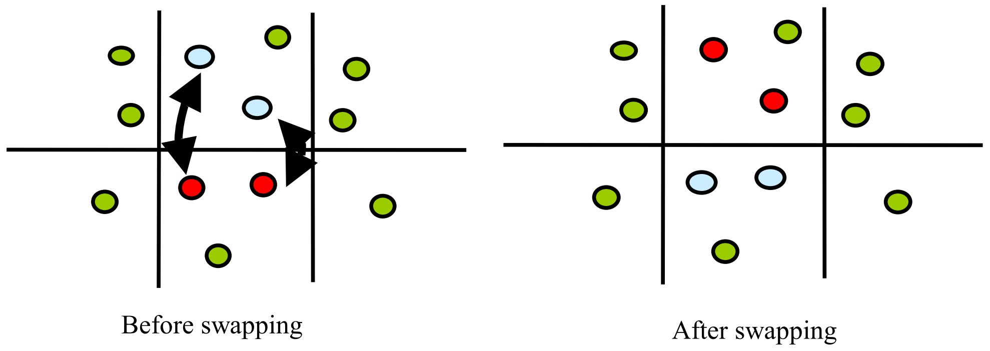


Figure 3-7. Vertical motion

3.3 Rendering TSV[4]

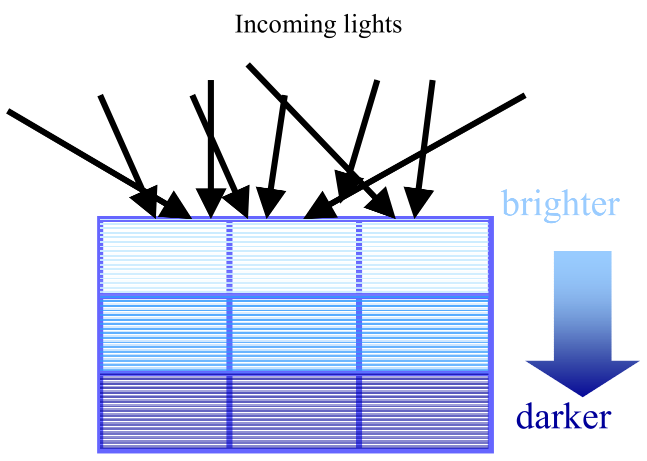
Each hair strand is reconstructed by connecting its corresponding particles. A straight line can be used to connect particles, but lines can generate sharp corners that seldom occur in real hair. Lagrange interpolation curves are used for the particle connections.

3.3.1 Probabilistic Shadow Generation

A simple shadow generation algorithm based on the TSV model is used, similar to the one used in [Kong99]. For shadow generation, lighting is assumed diffuse and omnidirectional. The inner hairs are considered more likely to be in shadow than the outer hairs. We propagate the probability of shadow starting from the outer cells. Let $\Phi_{i,j,k}$ represent the amount of shadow in cell $V_{i,j,k}$,

$$\Phi_{i,j,k} = \begin{cases} 0.0 & \text{, if } k = L \text{ (outermost cell)} \\ \Phi_{i,j,k+1} + Th \cdot \rho(V_{i,j,k+1}) & \text{, otherwise} \end{cases}$$

Here, *Th* is a constant to control the optical thickness of each hair.[5] The opacity function $\rho(V_{i,j,k})$ is proportional to the number of the particles in the cell. We vary the shadow values for each particle



[4] More recent versions of hair rendering algorithms are fully discussed in chapter 6.
[5] For example, thicker hairs will generate more shadow (occlude more light) than thinner hairs.

inside $\boldsymbol{V}_{i,j,k}$ by adding a small noise to $\boldsymbol{\Phi}_{i,j,k}$. Note that these shadow effects only change when combing or animation changes the particle distributions. A change in lighting or viewpoint does not affect these values.

3.3.2 Shading and Antialiasing

For each hair strand's local shading, we use the local illumination model developed by Kajiya and Kay [Kajiya 1989]. The details of the illumination models are discussed in chapter 5. Each hair strand is broken into a series of connected fine lines usually with an image length of 5-10 pixels. We calculate the result of the local shading for each node in hair strand. The shading value is multiplied by the shadow value. Each line segment is drawn using the standard Z-buffer algorithm.

Since hair is a very thin object when viewed under normal viewing conditions, one should be careful to reduce the artifacts from sampled line drawing. We employ supersampling by the accumulation buffer method. The hair image is drawn N times, each time shifted slightly in the image plane. Each image is scaled by 1 / N and added to the accumulation buffer. To avoid repeated evaluation of curves, a display list is prepared from a set of lines with associated colors based on the shadow and shading model. Line drawing is usually very fast if hardware acceleration is used.

3.4 Implementation and Results

The user first roughly designs the desired hairstyle with a set of surfaces. The Coons Patch is used as the representation for the reference surface (see appendix 3.A). Figure 3-8 shows a hairstyle designed with 4 bilinear Coons Patches. Figure 3-9 shows the actual combing function applied to a patch. Figure 3-9a shows the reference surface. Figure 3-9b shows a virtual comb with 20 teeth. The path of each tooth is modeled as a Lagrange Interpolation Curve. The paths are drawn as yellow curves. Figure 3-9c shows 4000 initial hair strands for the TSV model. Figure 3-9d shows 4000 hair strands after combing.

TSV model can be also considered as an animation tool for global control of hair motion. A global animation for each thin shell volume can be generated with simply deforming the reference surface. Furthermore, it is possible to add a simple dynamics to the reference surface. One can think of the hairs inside the TSV as constrained hairs due to our hair-hair interaction. One can also add free hairs that are unconstrained. As an animation test, we prepared two sets of hairstyles and performed linear interpolation (morphing) between them. Figure 3-10 shows a set of frames taken from the animation. The complete animations can be seen in http://graphics.usc.edu/~taeyong/tsv.html.

Figure 3-11 shows the effect of changing the comb parameters (7000 hairs were used). In our implementation, each path of the tooth is specified as a curve in the ***st*** plane. Currently, the combing functions are created and processed as batch operations. The number of teeth increases from bottom to top. In the top row, 40 teeth were used. In the middle row, 20 teeth were used. In the bottom row, 12 teeth were used. The width of each tooth was scaled accordingly so that neighboring teeth do not interfere. The depth of each tooth was set to roughly half the thickness of the volume. Small random variations were added to the path of each tooth.

The method was implemented on an SGI Onyx R10000 (only a single CPU and graphics pipe is utilized). It takes about 20~30 seconds to comb and generate the model shown in Plate2 (4000 hairs). Rendering is fast (1 – 2 seconds per frame). All the images shown are rendered at 512 x 512 pixel resolution. We used the resolution of 20x20x10 for all the TSVs in our tests. A large fraction of the computing time is consumed in the reconstruction of the hair strand from the TSV particles due to the numerous evaluations of the Coons Patch. For local lighting of the test scenes, two lights were used for the diffuse and specular shading.

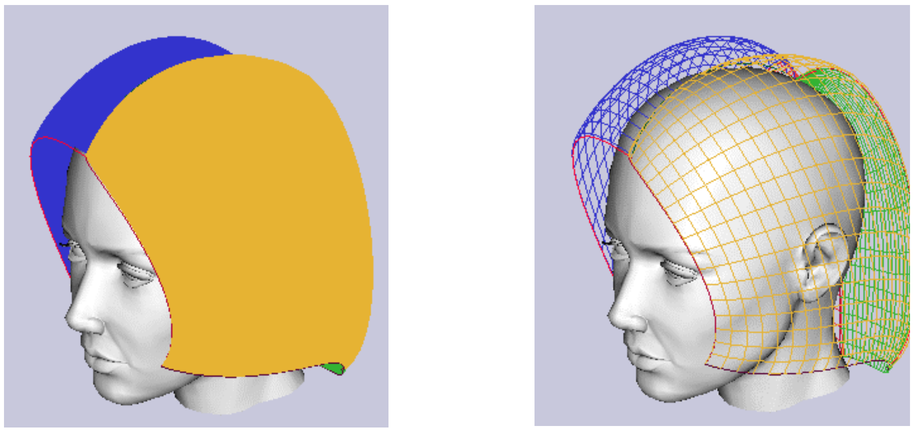
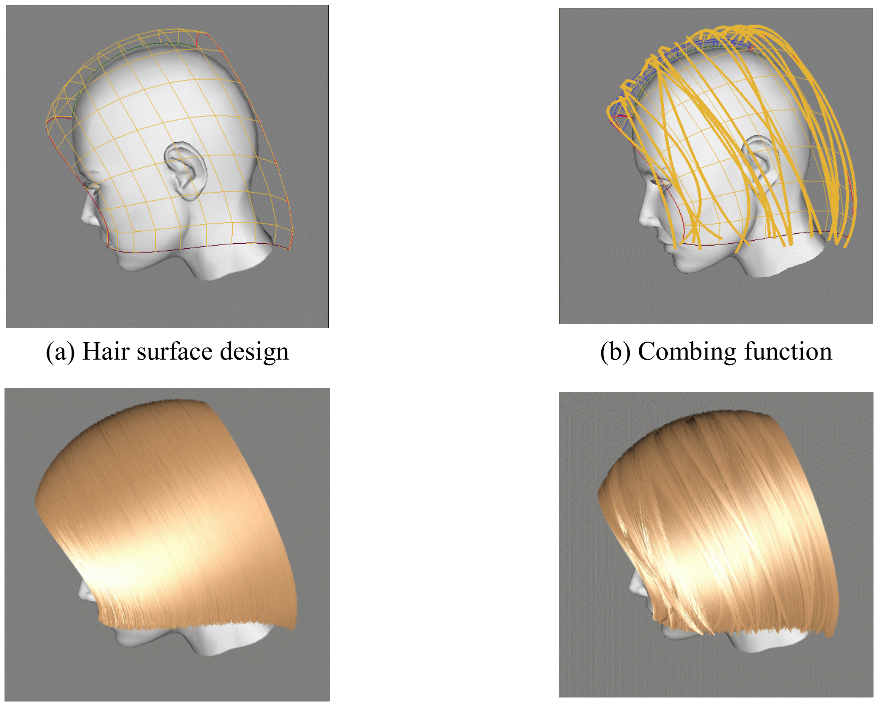


Figure 3-8. A hair design with 4 patches (Each patch has a different color).



(a) Hair surface design

(b) Combing function

(c) Hair model before combing

(d) Combed Hair

Figure 3-9 Result of hair combing

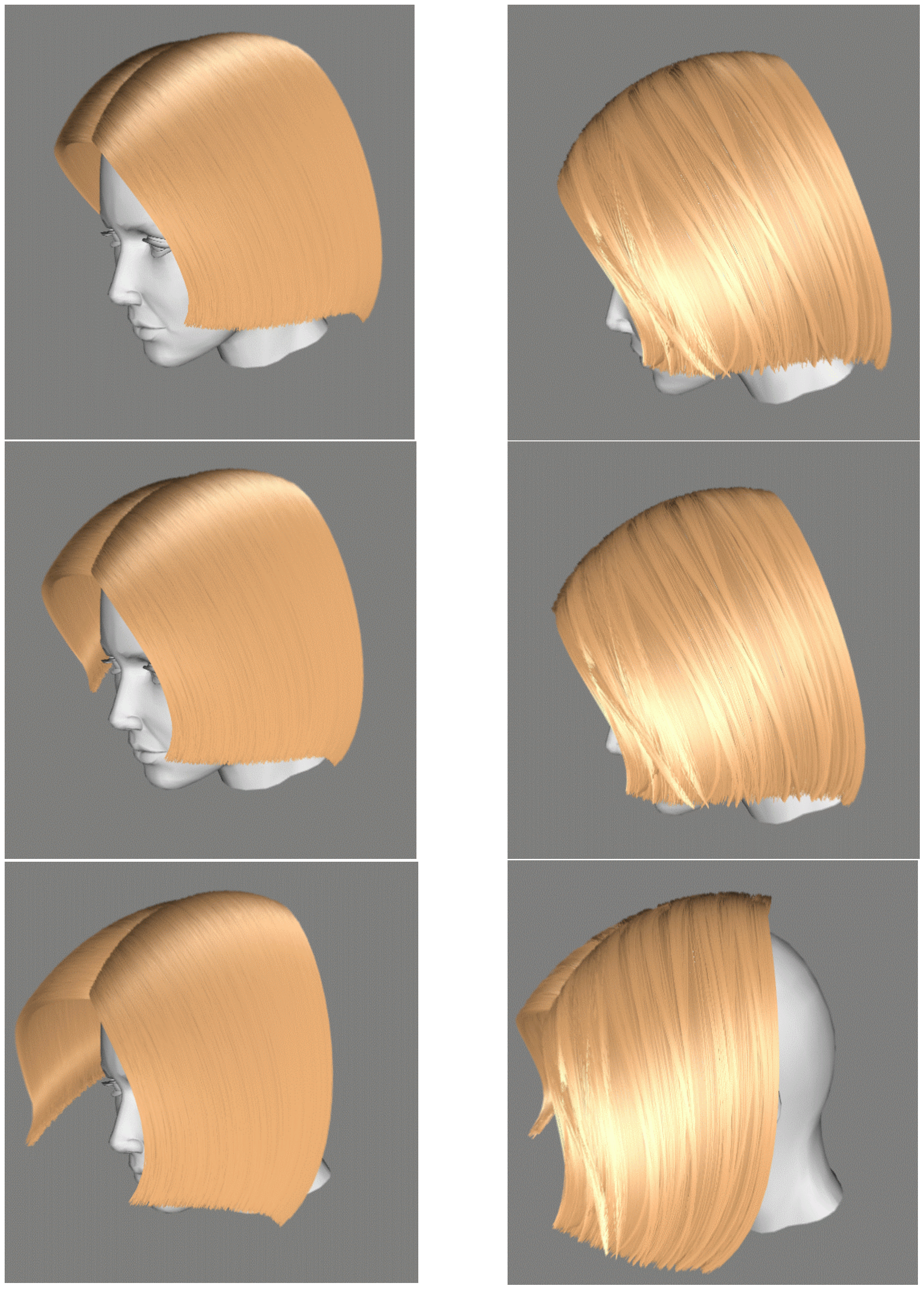


Figure 3-10. Animation of TSV. The left column shows hairs that were not combed, and the right column shows *combed* hairs.

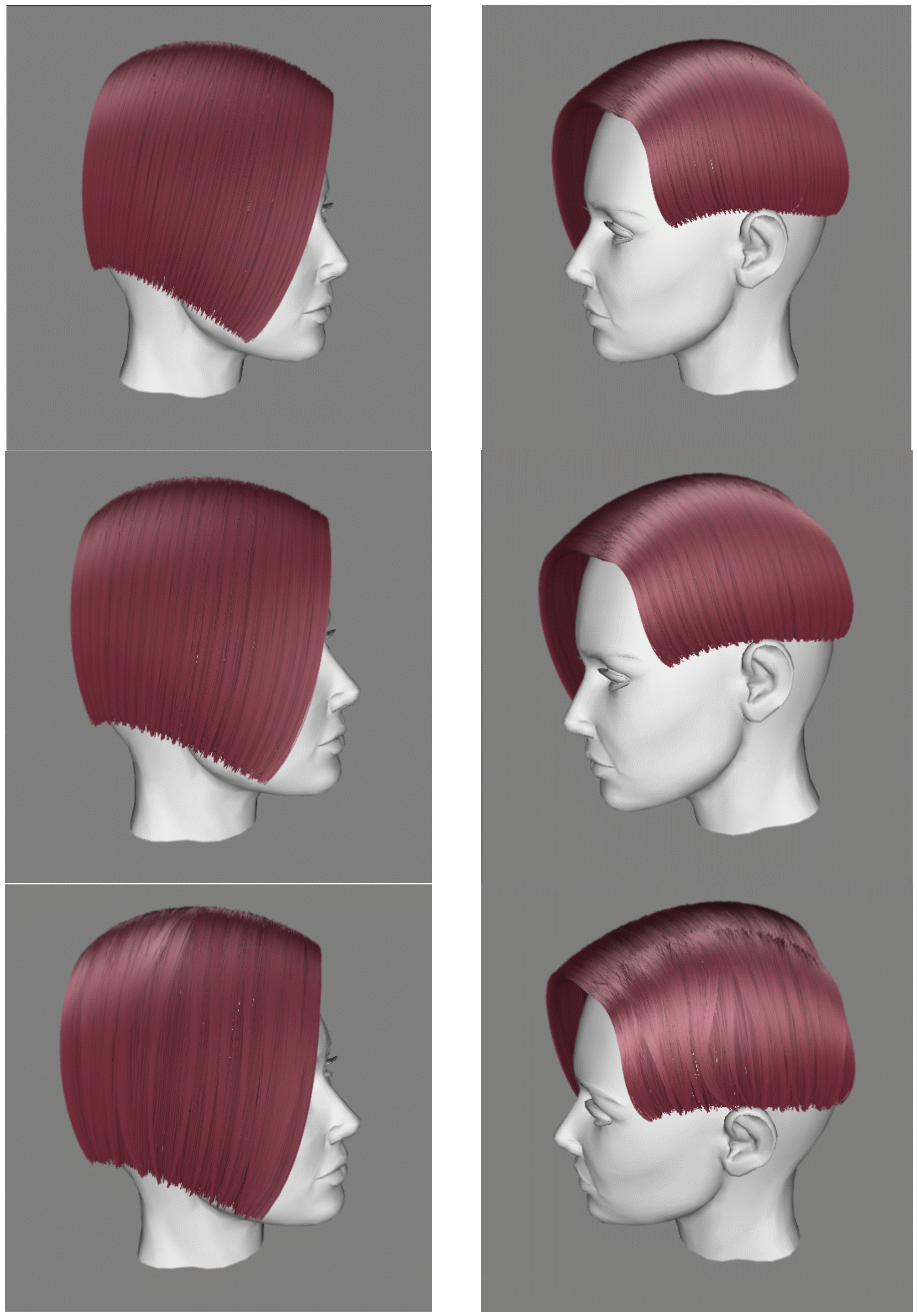
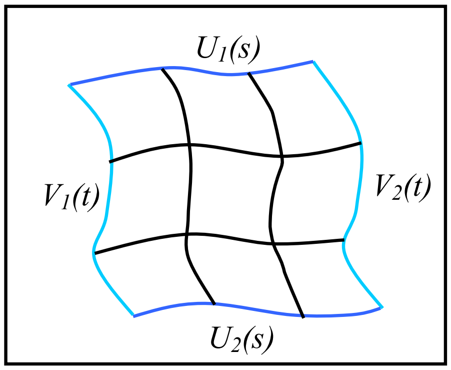


Figure 3-11. The effect of different combing functions.



A Coons Patch

3.A Coons Patch

A Coons Patch is defined by 4 boundary curves [Coons 1967]. These curves can be any parametric curve as long as their endpoints are connected to form a closed boundary. Let us say the 4 boundary curves are: $U_1(s)$, $U_2(s)$, $V_1(t)$, $V_2(t)$.

Any point $P = S(s,t)$ is calculated as follows.

$P = S(s,t) = (1 – t)\ U_1(s) + tU_2(s) + (1-s)\ L(t) + s\ R(t)$

, where

$L(t) = V_1(t) – (1-t)\ V_1(0) - tV_1(1)$

$R(t) = V_2(t) – (1- \quad t)\ V_2(0) – tV_2(1)$

Chapter 4

4. Interactive Multiresolution Hair Modeling and Editing[*]

The TSV method in the previous chapter is limited to mostly flat and surface like hairstyle. With the method, it is difficult to model more volumetric hairstyles such as curly hair. In this chapter, a multiresolution hair modeling (MHM) system is introduced. In this system, a hair cluster, a group of hairs, is represented with a generalized cylinder (GC). The cluster representation greatly reduces the number of parameters that the user has to specify to design a hairstyle. However, a single cluster may not be flexible enough for modeling every possible variation in hair shape. If the user attempts to model detailed hairstyle with large number of clusters, the modeling task becomes enormously time consuming. For example, a ponytail hairstyle made of many curly hair clusters can be extremely difficult to model with manual editing of individual cluster. It can be convenient to first design the shape of ponytail at large scale, and to specify the clusters inside the ponytail hairstyle to be curly. MHM aims at achieving this goal.

MHM extends the idea of the TSV method in that overall shapes and individual details are separated. In the TSV model, overall shape is designed with a parametric patch while details are specified with virtual hair combing functions. In MHM, generalized cylinders are used at multiple levels to design both the overall shape and detail. Figure 4-1 illustrates an example multiresolution hair modeling procedure. An overall hairstyle is designed with a small number of generalized

[*] Presented in ACM SIGGRAPH 2002

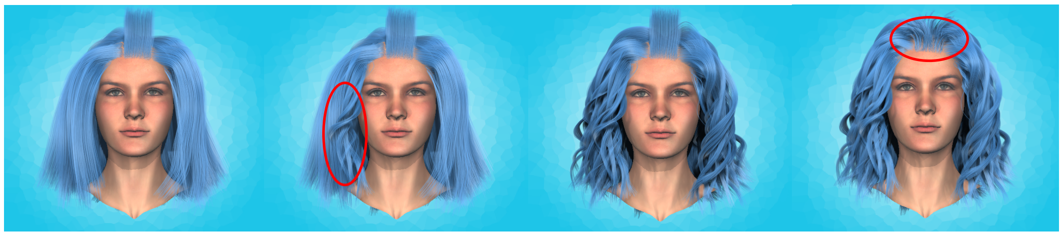


Figure 4-1. An example multiresolution hair design procedure. Each hair model results from interactive multiresolution editing operations. Left to right: 1. The user roughly designs a hairstyle with about 30 high-level clusters. 2. One hair cluster (inside the ellipse) is subdivided and made curly. 3. The curly cluster is copied onto other clusters. 4. The bang hair cluster (inside the ellipse) is subdivided and refined and the model is further refined.

cylinders. To add more detailed variation, a hair cluster is subdivided and this subdivided cluster is edited to be curly by changing the twist parameter of the generalized cylinder. This user-designed curly cluster is then copied onto other clusters with copy-and-paste tool. The last image in Figure 4-1 shows the final hair model after further refinements.

The multiresolution concept is not new in computer graphics. To model complex objects, researchers have successfully exploited multiresolution concepts for continuous curves, surface editing, and volume sculpting (see [Stollnitz 1996] for examples). Their major benefit is the user's freedom to choose the appropriate level of detail for a desired model manipulation. In this spirit, one could formulate a multiresolution framework for hair modeling. The result is the Multiresolution Hair Modeling (MHM) system presented in this chapter. However, hair models are inherently volumetric and contain a large number of disjoint hair strands. Thus, MHM differs from other multiresolution techniques in applying the concept of multiresolution to hair. In the context of hair modeling, multiresolution manipulations are achieved with a hierarchy of generalized cylinders. MHM allows users to interactively move, scale, curl, and copy a portion of an evolving hair model at any level of detail.

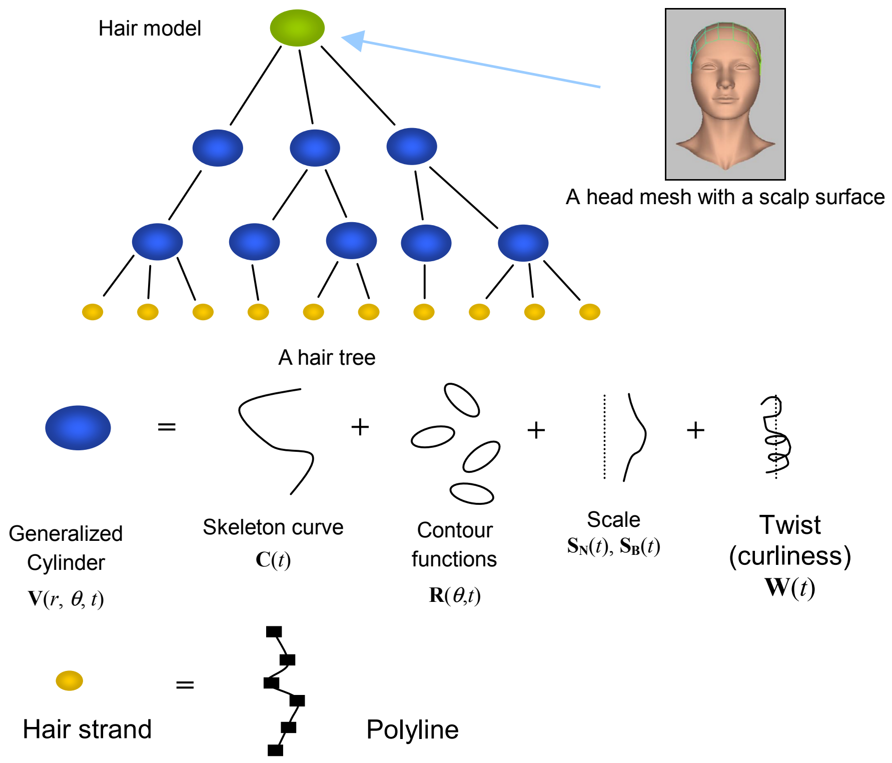

Hair model
A head mesh with a scalp surface
A hair tree
=
+
+
+
Generalized Cylinder $\mathbf{V}(r, \theta, t)$
Skeleton curve $\mathbf{C}(t)$
Contour functions $\mathbf{R}(\theta,t)$
Scale $\mathbf{S}_N(t)$, $\mathbf{S}_B(t)$
Twist (curliness) $\mathbf{W}(t)$
Hair strand
=
Polyline

Figure 4-2. Graphical view of a multiresolution hair model.

4.1 Overview

Figure 4-2 illustrates the structure of a multiresolution hair model. A parametric patch on the scalp surface defines the region where hair can originate. A hair model is constructed hierarchically, starting from a small set of generalized cylinders (GCs). A GC defines the boundary of each hair cluster, controlling a group of clusters or hair strands. The user interactively subdivides each hair cluster, adding more detail until the desired appearance is achieved. Subdivision steps add new nodes to the hierarchy called a hair tree. Editing operations such as curling, moving, copying, and selecting are applied to nodes of the hair tree. While the user edits a hairstyle, the system interactively visualizes the edited model using various rendering options (the details of hair rendering algorithms are presented in chapter 6).

4.2 Hair Cluster Representation

A GC defines the boundary of a hair cluster. In the following sections, the term GC and hair clusters are used interchangeably.

4.2.1 Generalized Cylinder

A GC is defined with a skeleton curve **C**(*t*) and a set of contours placed along the curve (Figure 4-3).

$$\mathbf{GC}(\theta, t) = \mathbf{C}(t) + \mathbf{R}(\theta, t) \qquad (4\text{-}1)$$

where **C**(*t*) is a space curve parameterized by *t* and **R**(*θ*, *t*) is a contour function centered at **C**(*t*). *θ* is an angle around the principal axis of a reference frame. Similar definitions can be found in [Aguado et al. 1999; Xu and Yang 2000]. Alternatively, a GC can be viewed as rotational sweep of a profile curve [Kim et al. 1994].

With an additional parameter *r*, any point around the GC can be defined as

$$\mathbf{V}(r, \theta, t) = \mathbf{C}(t) + r\,\mathbf{R}(\theta, t) \qquad (4\text{-}2)$$

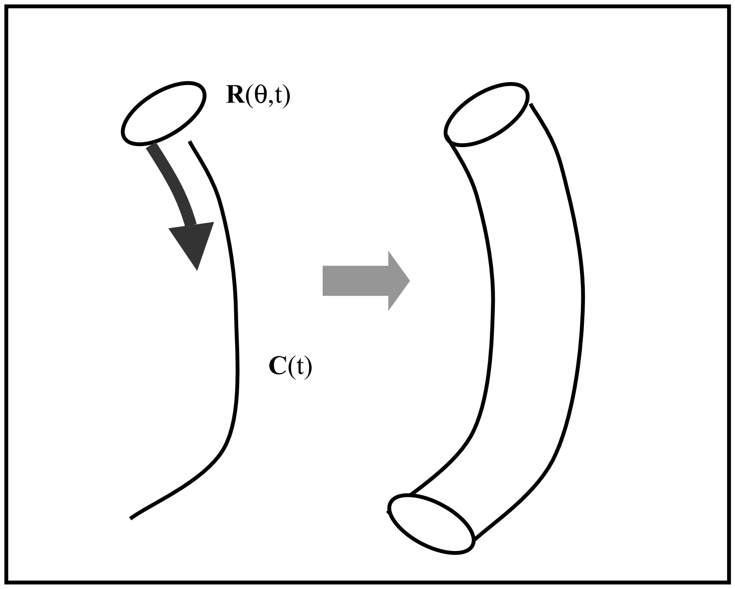


Figure 4-3. Creating a generalized cylinder

Constructing a generalized cylinder requires each contour to be aligned properly with its neighbors so that the structure does not twist. The alignment is usually provided with a reference frame, a point and three orthonormal vectors that define position and orientation along the central axis of the cylinder. Figure 4-4 illustrates two generalized cylinders generated using different frames. The middle one is generated with a constant frame, while the right one is generated with correctly rotating frames. Note that shapes can be distorted at highly curved region with constant frame.

The Frenet frame [Bloomental 1990] is convenient because it can be analytically computed at arbitrary point along the space curve. The frame uses a tangent vector **T**, and the principal normal **N**, and the binormal **B**. **T** is the first derivative of the curve. **K** is the direction of curvature such that $\mathbf{K} = \mathbf{V} \times \mathbf{Q} \times \mathbf{V} / |\mathbf{V}|^4$, where **Q** is the second derivative of the curve.

Thus, $\mathbf{N} = \mathbf{K} / |\mathbf{K}|$, $\mathbf{B} = \mathbf{T} \times \mathbf{N}$.

There are two problems with the Frenet frame, as pointed out in [Bloomenthal 1990] and [Yang 2000]. First, the frame is undefined when the curve is straight or on the inflection point. Moreover, the curvature vector can suddenly reverse direction on either side of an inflection point, inflicting a violent twist in a progression of Frenet frames.

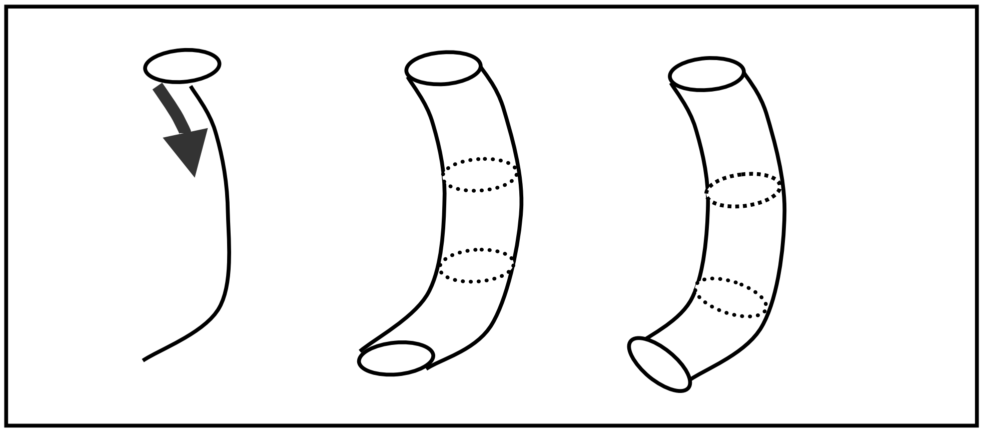


Figure 4-4. Effects of different reference frames.

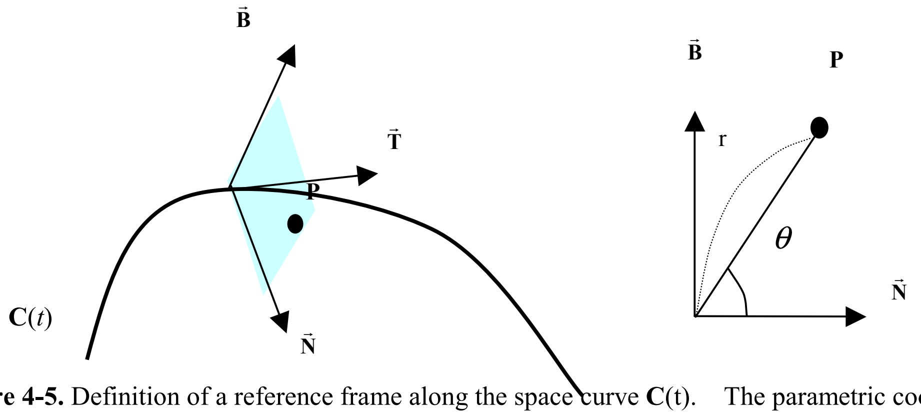


Figure 4-5. Definition of a reference frame along the space curve **C**(t). The parametric coordinate for a point **P** is given as (r, θ, t), where θ is defined as the angle around the tangent vector $\vec{\mathbf{T}}$, starting from the principal normal $\vec{\mathbf{N}}$.

The rotation minimizing frames method improves upon the Frenet frame (see [Bloomenthal990]). The idea behind rotation minimizing frames is to define an initial reference frame at the beginning of the curve and then propagate the frame along the curve using small, local rotations. The first frame is generated using the tangent vector **T** at the beginning, and two vectors **N** and **B** from the contour **R**(θ, t). **N** is in direction of $\theta = 0^o$ and **B** is in direction of $\theta = 90^o$. Both vectors lie on the plane of the contour.

A set of frames $\mathbf{F_i}$ ($1 \leq i \leq N$) is calculated along the curve **C**(t). Using these pre-computed frames, any frame on an arbitrary point on the curve can be interpolated. The pre-computed frames are linearly interpolated to form a smoothly shaped generalized cylinder. Let $\vec{\mathbf{T}}(t)$, $\vec{\mathbf{N}}(t)$, and $\vec{\mathbf{B}}(t)$ denote the interpolated frame vectors at t ($0 \leq t \leq 1$) (Figure 4-5). The world coordinate of a point parameterized by (r, θ, t) is

$$\mathbf{V}(r,\theta,t) = \mathbf{C}(t) + r\,\mathbf{R}(\theta,t)\left\{\cos(\theta)\overrightarrow{\mathbf{N}}(t) + \sin(\theta)\overrightarrow{\mathbf{B}}(t)\right\} \tag{4-3}$$

4.2.2 Representation of Contour

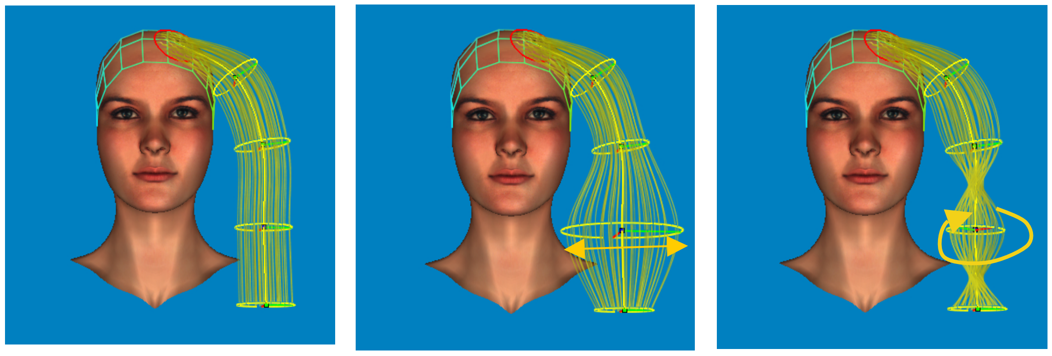
$\mathbf{R}(\theta)$ is a contour shape function dependent on θ. $\hat{\mathbf{R}}(\theta)$ is defined as an offset distance function from a circle with a global radius s ($\mathbf{R}(\theta) = s + \hat{\mathbf{R}}(\theta)$). N pairs of angles and offset radius values control the shape of a contour along with the global radius s. The current implementation uses $\theta_i = 2i\pi / N$ and $\hat{\mathbf{R}}(\theta)$ is computed by linear interpolation of the angle / offset pairs. By definition, the contour is star-shaped, which is found to be flexible enough to represent the boundary of hair clusters. The offset function can efficiently represent star-shaped contours with a small set of parameters (currently, $N = 32$ is used). The continous contour function $\mathbf{R}(\theta, t)$ is linearly interpolated from a discrete number of contours.[6]

4.2.3 Auxiliary Functions

Adding auxiliary scale and twist terms completes the description of GC. The scale terms $\mathbf{S}_\mathbf{N}$(t) and $\mathbf{S}_\mathbf{B}$(t) let the user easily control the stretching and shrinking of GCs along the two axis vectors $\vec{\mathbf{N}}$ and $\vec{\mathbf{B}}$, while direct modification of the contour functions allows fine detailed control over the shape. The twist term $\mathbf{W}$(t) controls the curliness of a hair cluster. Adding these terms yields

$$\mathrm{V} = \mathrm{C}(t) + r\,\mathrm{R}(\theta, t)\left\{\cos(\theta + \mathrm{W}(t))\mathbf{S}_\mathbf{N}(t)\vec{\mathbf{N}}(t) + \sin(\theta + \mathrm{W}(t))\mathbf{S}_\mathbf{B}(t)\vec{\mathbf{B}}(t)\right\} \quad (4\text{-}4)$$

Figure 4-6 shows examples of changing these auxiliary functions.



(a) (b) (c)

Figure 4-6. Effects of auxiliary terms. (a) Original cluster (b) Scale change (c) Twist change.

[2]There exist more rigorous methods to define and interpolate the contours of GCs [Aguado 1999]. The definition of contours here aims at simplicity and computational efficiency.

4.3 Scalp Surface

A *scalp surface* is a parametric patch defining the region of a head where the user can place hair. The scalp surface is constructed once for each head model and it is useful in many ways. 1) The user can easily place and move the GC contour on the scalp mesh. 2) Sampling and assigning hair strands become simpler. 3) The subdivision of hair clusters is reduced to a 2D problem. 4) The scalp parameterization provides the user with convenient means to select clusters in 2D space. The current system uses a tensor-product Catmull-Rom spline patch that the user wraps over the head model. The user interactively specifies each control point of the spline patch (Figure 4-8). Alternatively, one could also use polar coordinate mapping as in [Rosenblum 1991], [Yu 2001].

Scalp space (middle figure in Figure 4-7) is spanned by *u* and *v*, the parametric coordinates of the surface. *World space* is specified by the 3D world coordinate frame. Let $\mathbf{R}^s(\theta)$ denote the contour function in scalp space and $\mathbf{R}^w(\theta)$ denote contours in world space. Scalp space contours are used for hair strand generation and for the initialization of multiple world space contours that define the shape of a GC.

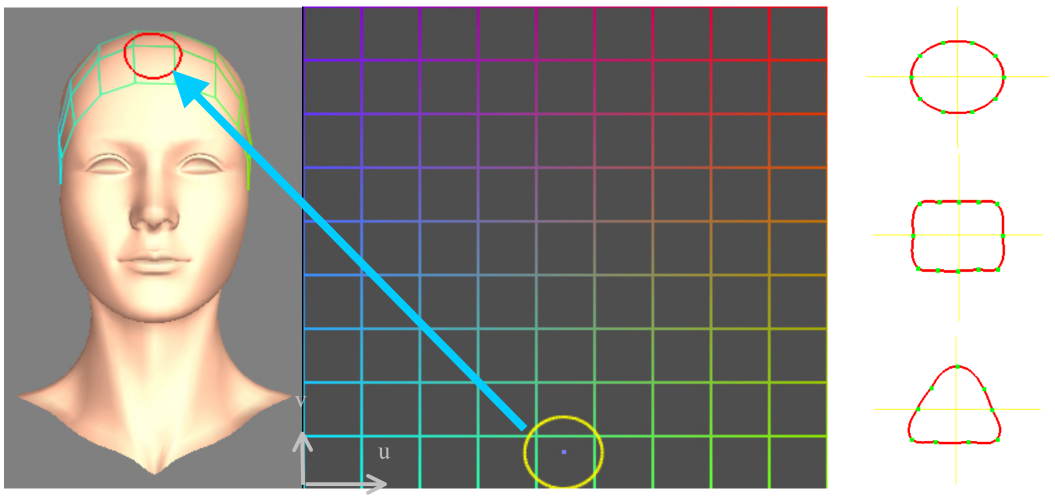


Figure 4-7. Placing a contour on the scalp. The user selects a contour (right) and interactively places it in 2D scalp space (yellow circle in the middle), which defines its 3D position (red circle on the left). The grid is color coded to help the user match corresponding positions.

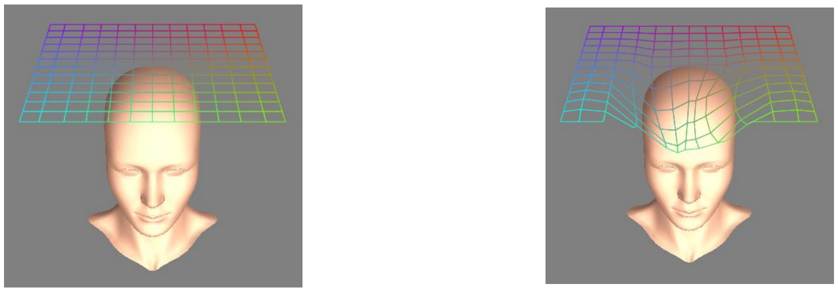


Figure 4-8. Construction of a scalp surface. 13 x 13 control points are used. The user clicks over the head mesh to specify the 3D position of each control point. This interactive process takes about 20 minutes.

When the user places a 2D contour on the scalp, the corresponding GC is created. Let the scalp surface is parameterized as **S**(u,v). First, the root position $\mathbf{P_0}$ of the skeleton curve is calculated as $\mathbf{P_0} = \mathbf{C}(0) = \mathbf{S}(u_c, v_c)$, where u_c and v_c denote the center of the scalp space contour. Let $\mathbf{N_0}$ be the surface normal at $\mathbf{P_0}$. Initially, the skeleton curve is formed as a straight line (each control point $\mathbf{P_i}$ of the skeleton curve is given as $\mathbf{P_i} = \mathbf{P_0} + iL / (N - 1)\, \mathbf{N_0}$, where N is the number of control points specified by the user) and L is the length of the skeleton curve. As a next step, we convert $\mathbf{R}^s(\theta)$, the scalp space contour to $\mathbf{R}^w(\theta)$ in world space. Let $\mathbf{R_i}$ be the 3D position of each sample point of the contour (recall from section 4.2.1 that each contour is represented with the sample values $\mathbf{R}(\theta_i)$).

$$\mathbf{R_i} = \mathbf{S}(u_c + \mathbf{R}^s(\theta_i)\cos(\theta_i), v_c + \mathbf{R}^s(\theta_i)\sin(\theta_i)) \tag{4-5}$$

To make the contour planar, we project $\mathbf{R_i}$ to the plane formed by $\mathbf{P_0}$ and $\mathbf{N_0}$. Let $\hat{\mathbf{R}}_i$ be such a projected point. Then, $\mathbf{R}^w(\theta_i)$ is the distance from the center point $\mathbf{P_0}$ to $\hat{\mathbf{R}}_i$ in 3D space.

$$\mathbf{R}^w(\theta_i) = \left\| \hat{\mathbf{R}}_\mathbf{i} - \mathbf{P_0} \right\| \tag{4-6}$$

The contour function is then decomposed into the global scale *s* and offset function as defined in section 4.2.1. The number of contours is the same as the number of control points of the skeleton curves. These contours are simply copied at each control point of the skeleton curves.

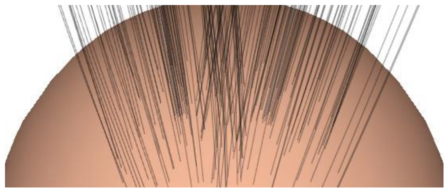
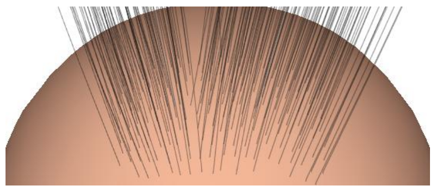
(a)  (b) 

Figure 4-9. Hair strand generation schemes. (a) Independent hair generation for each cluster causes uneven hair density in overlapping regions. (b) By assigning pre-sampled hair strands to clusters, the overlap problem is solved.

4.4 Generation of Hair Strands

Hair strands are represented as polylines (a set of connected line segments). Using the generalized cylinder equation of (4-4), one can creating hair strands independently within each GC (Figure 4-9a). However, this can cause visually distracting hair density variation since we allow arbitrarily shaped scalp space contours that can overlap each other (Figure 4-10). Rather than trying to prevent the overlaps (which might be very costly), we sample the root positions of hair strands first, and assign the strands to their *owner* GCs (Figure 4-9b).

Initially, (or whenever the total number of hair strands changes), the root position (u_h,v_h) of each hair strand is uniformly sampled on the scalp grid. For each hair strand, we determine if there exists a GC that owns the strand. If there is such a GC, the parametric coordinate (r_h, θ_h) of the strand is computed as follows.

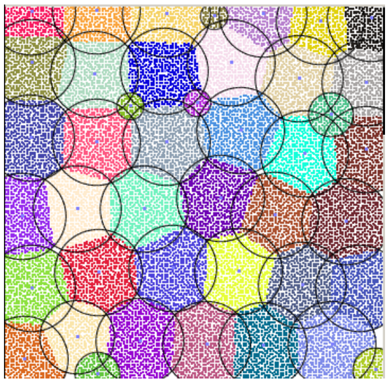


Figure 4-10. Scalp space contours can overlap each other. Colored dots illustrate the root positions of strands. Each color indicates the owner cluster of hair strands.

A hair strand is given its location on the scalp by (u_h,v_h). Let the center of the contour $R^s(\theta)$ of each cluster be represented by (u_c,v_c). Let $\Delta u = u_h - u_c$ and $\Delta v = v_h - v_c$. Then, the angle θ_h is given as

$$\theta_h = \cos^{-1}\left(\frac{\Delta u}{\sqrt{\Delta u^2 + \Delta v^2}}\right) \tag{4-7}$$

,and $\theta_h = 2\pi - \theta_h$ if $\Delta u < 0$

A hair strand is contained by a cluster if

$$\mathbf{R}^S(\theta_h) \le \sqrt{\Delta u^2 + \Delta v^2}$$

A bounding box of each contour is used to speed up the test. Also, the inverse cosine function is tabulated. If a hair strand is assigned to a cluster, the parametric coordinate for the root position of the hair strand is given as (r_h, θ_h), where

$$r_h = \frac{\sqrt{\Delta u^2 + \Delta v^2}}{\mathbf{R}^s(\theta_h)}$$

, and θ_h is given in (4-7). Given (r_h, θ_h), the hair strand is created by connecting points computed with equation 4, incrementing t from 0 to 1.

Note that equation 4-4 assumes that each GC contour lies in the plane perpendicular to the skeleton curve. Since the scalp shape is curved, planar contours can misalign the root positions of strands. To handle the problem, the root position ($t = 0$) of each hair strand is directly evaluated on the scalp, using

$$\mathbf{V}(r_h, \theta_h, 0) = \mathbf{S}(u_h, v_h) \tag{4-8}$$

,where $\mathbf{S}(u,v)$ is the parametric scalp patch.

However, this modification can still cause unnatural bending along hair strands if other points of polyline segments are directly evaluated from equation 4-4 (Figure 4-11a). Similar problems can arise from self-intersections in GCs of high curvature (Figure 4-12a). Using an interpolating Catmull-Rom spline curve, each hair strand is guaranteed to be smooth again (Figure 4-11b, Figure 4-12b). The number of spline control points is the same as that of the skeleton curve of the strand's owner GC. The polyline segments of each hair strand are evaluated from the spline interpolation.

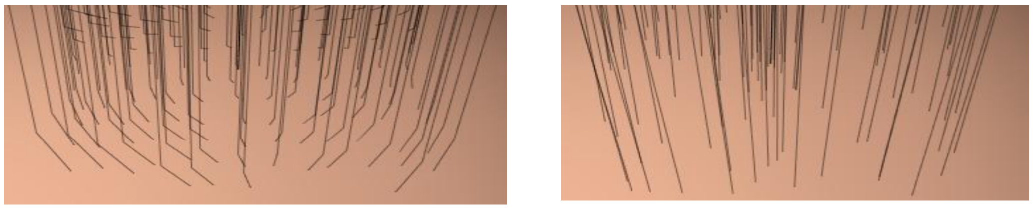

(a) (b)

Figure 4-11. Guaranteeing the smoothness of a hair strand. (a) Straight line connections can cause unnatural bending around the root position and in highly curved region. (b) Using spline interpolation at strand-level, smoothness is guaranteed.

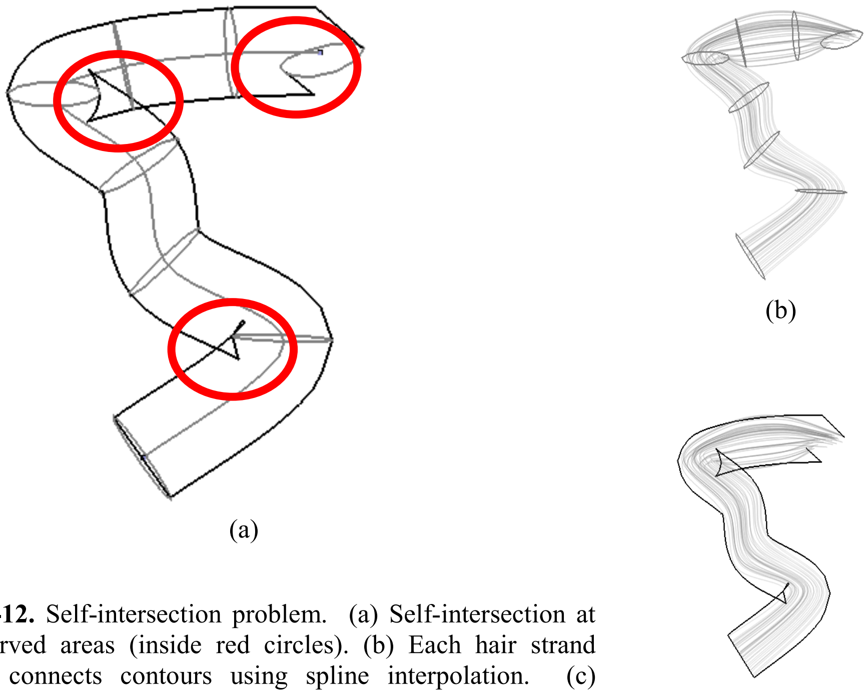

(a) (b) (c)

Figure 4-12. Self-intersection problem. (a) Self-intersection at highly curved areas (inside red circles). (b) Each hair strand smoothly connects contours using spline interpolation. (c) Comparison of hair strands with the boundary of the GC.

Varying the length of each hair strand increases the model's natural appearance. A random sample η is taken from a uniform distribution from 1-k to 1 ($0 \leq 1\text{-}k \leq \eta \leq 1$), where k denotes the user specified degree of tip length variation. Adding the length variation, we evaluate the spline controls points of each hair strand using ηt instead of t for equation (4-4).

4.5 Subdivision

When a parent cluster is subdivided, contours and skeleton curves of its child clusters are created. The contours of child clusters are inherited from those of the parent cluster unless the user specifies alternative shapes. Inherited contours are scaled by $\rho / \sqrt{N}$, where N is the user-specified number of child clusters and ρ controls the degree of overlap between child clusters. For world space contours, ρ is set to 1.0. For scalp space contours ρ is set to a value larger than 1.0 (typically $1.3<\rho<1.5$), to ensure that the subdivided region is fully covered by the contours of child clusters.

To place the child clusters on the scalp, we first sample random positions inside the contour of their parent. This often produces uneven distributions of contour positions (Figure 4-13a). We use the position relaxation algorithm by [Turk 1991] to improve the distribution (Figure 4-13b). Given these positions, skeleton curves are created using the method described in section 4.4.

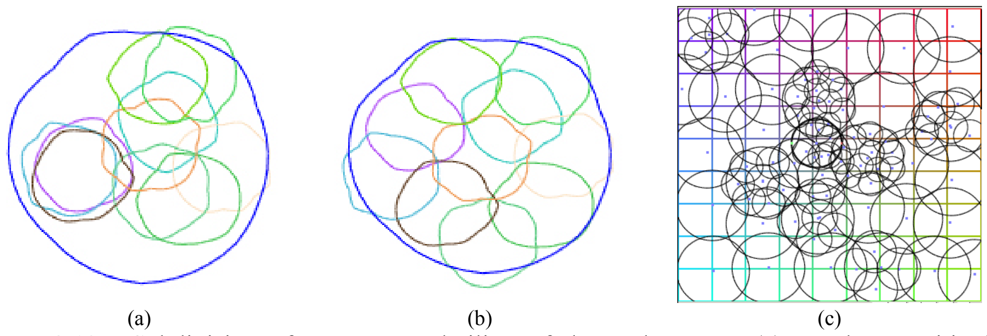

(a) (b) (c)

Figure 4-13. Subdivision of contours and tiling of the scalp space. (a) Random positioning (ρ=1.4). (b) Position relaxation. (c) An example tiling of the scalp surface after a few subdivision steps.

4.5.1 Deciding the Owners of Hair Strands

After child clusters are positioned on the scalp, hair strands are re-assigned to the new clusters. As shown in Figure 4-13c, contours of the child clusters can overlap with each other or contours of existing clusters. In overlapping regions, we assign hair strands to a cluster with the closest center position in scalp space, similar to computing the centroidal voronoi diagram [Hausner 2001] of the center positions of all the clusters on the scalp (Figure 14a). However, simply computing the voronoi diagram may be problematic when precise controls for hair grouping are desired (e.g., for parting).

Our hair ownership decision algorithm shown in Figure 4-15 allows the user to design and tune the tiling and subdivision patterns if necessary (Figure 4-14b). For each hair strand, the algorithm traverses the hair tree starting from root clusters. The closest cluster containing the strand is chosen at each depth. The iteration is repeated until it reaches a leaf cluster. If there is only one such leaf cluster, the strand is assigned to the cluster regardless of the distance to other

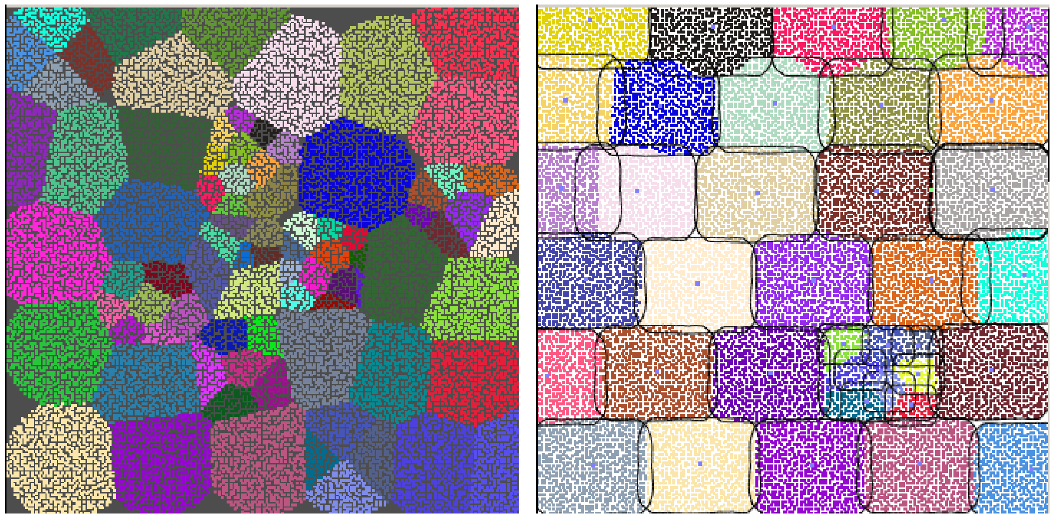


Figure 4-14. Hair strand assignment patterns. a) Hair strand assignments based on contours shown in Figure 4-13c. The root positions of hair strands are drawn as colored dots. (b) User controlled tiling and subdivision. Note the clear borders between contours.

clusters.[7] When two or more leaf clusters contain the same strand due to overlaps, the cluster with a center position closest to the strand is selected. When a hair strand is contained in a non-leaf cluster, but not contained in any of its descendents[8], the strand is assigned to the closest descendent leaf cluster. Note that we allow only leaf clusters to own hair strands, and for each hair strand, there is at most one owner. Thus, we can maintain consistent hair density and limit the total number of hair strands; both features prove useful while level-of-detail manipulations control the appearance of the overall hair model.

```
function FINDOWNEROFAHAIRSTRAND (HairStrand H)
O ← NULL ,  C ← first root cluster,  done ← FALSE
while (!done and C != NULL) do
    done ← TRUE
    forall {S | S is a sibling of C or C itself } do
        CHECKFORINCLUSION(H,S)
        if S contains H and its center is closer to H than O
            O ← S
            done ← FALSE
    if (!done)
        C ← O.FIRSTCHILD
if (O is not a leaf cluster)
    forall { L | L is a descendent of O and a leaf node } do
        if L's center is closer to H than O
            O ← L
        return O
```

Figure 4-15. Hair ownership decision algorithm

[7] In contrast, a voronoi diagram based approach may assign the strand to another cluster even if the hair strand does not originate from the region bounded by the cluster.
[8] This case occurs when the contours of child clusters do not perfectly cover the contour of the parent cluster even after the position relaxation step. Note that we set ρ to a value larger than 1.0 to minimize this artifact.

4.6 Interactive Manipulation

The core of MHM lies in the user's ability to edit any node in the hair tree. The user controls the position, contours, scale, and twist of any GC, affecting the hair model at multiple levels of detail (section 4.6.1). A sub-tree in the hair tree can be used as a user-defined *style template*. The copy/paste tool (section 4.6.2) transfers a user-designed style from one cluster to other clusters. Section 4.6.3 describes a set of selection methods that exploit the scalp surface and the hair tree. During interactive manipulation, hair/head penetration is detected and avoided (section 4.6.4).

4.6.1 Editing Properties of Hair Clusters

The user interactively manipulates GC parameters such as scale, twist, and skeleton shape (equation 4). When editing leaf clusters, the system simply reassigns the parameters and update hair strands. When editing a non-leaf cluster, all of its descendent clusters must follow the shapechanges to preserve their relative position and orientation (Figure 4-16).

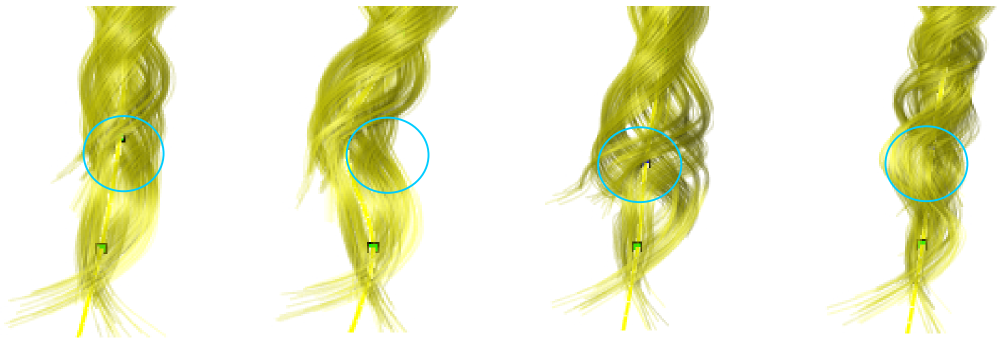

(a) (b) (c) (d)

Figure 4-16. Interactive manipulation of a hair cluster. (a) The user picks the control point (inside the blue circle) of a high-level cluster. The user can (b) move, (c) scale, and (d) twist the region around the control point.

When a non-leaf cluster is edited, all the descendant clusters are attached to the cluster as follows. Let **V** be the cluster before editing. Assume that the user changes **V** to **V'**. Then each descendant cluster is updated as follows. 1) For each control point **P** of the skeleton curve, we find the parametric coordinate (r,θ,t) of **P** with regard to the cluster **V** such that $\mathbf{P} = \mathbf{V}(r,\theta,t)$ (*bind*). 2) The new position **P'** is recalculated using $\mathbf{P'} = \mathbf{V'}(r,\theta,t)$ (*update*).

The bind procedure is the inverse transform of equation (4-4), *i.e.* $(r,\theta,t) = \mathbf{V}^{-1}(\mathbf{P})$. Given a point **P** and a cluster **V**, the parametric coordinate (r,θ,t) of the point with respect to the cluster is found as follows. First, t is chosen as the value that minimizes the Euclidean distance between point **P** and the skeleton curve **C**(t). Let $\mathbf{P_C}$ be such a point that $\mathbf{P_C} = \mathbf{C}(t)$. Then θ is given as the angle between a vector connecting **P** and $\mathbf{P_C}$ and the principle normal **N**(t). The angle should be corrected for the scaling terms $\mathbf{S_N}(t)$, $\mathbf{S_B}(t)$. Let the projection of the vector $\overrightarrow{P_C P}$ on **N**(t), **B**(t) be $\mathbf{P_N}$, $\mathbf{P_B}$. Then, the angle θ is given using equation (4-7), by letting $\Delta u = \mathbf{P}_N / \mathbf{S_N}(t)$, and $\Delta v = \mathbf{P_B} / \mathbf{S_B}(t)$. After correcting for the twist term **W**(t), $\theta = \theta - \mathbf{W}(t)$. The parameter r is the ratio between the Euclidean distance between **P**, $\mathbf{P_C}$ and $\mathbf{R}(\theta,t)$.

$$r = \|P - P_C\| / \mathbf{R}(\theta, t)$$

The bind process is performed whenever the user initiates an action (for example, selects a hair cluster and starts editing it), whereas the update process occurs at every step of the user interaction. Hair strands are considered *statically* bound to a cluster when their owner is decided. The bind/update method is similar to *Wires* [Singh 1998], but our method compensates for the axial scaling and twist terms, and no falloff term is used.

As reported in [Singh 1998], the closest point on a curve becomes ambiguous as a point moves away from the curve or if the curve is of a high curvature. This can cause the control points of skeleton curves to *drift* during the bind/update procedure. However, the bind/update procedure is only used for rough editing of the hair model, while the user eventually *corrects* the drift by refining child clusters. This problem could be obviated by fitting the entire skeleton curves of the bound clusters, not just the control points, to the skeleton curve of the binder cluster.

In the bind/update procedure, *every* descendent cluster is bound to the edited cluster, not just immediate child clusters. It is tempting to store the positions of child clusters as offsets to their parent cluster to speed up the bind operation. However, there are more time-consuming operations such as hair strands updates, temporary binding, and penetration avoidance. The offset approach could be inefficient for these operations that require the world coordinates of each cluster. For example, if all the root clusters are subdivided three times, the offset approach will incur three times more GC computations than the current approach.

Temporary/Local Binding

The bind/update process can be also used between clusters with no common ancestor (e.g., attaching a root cluster to another root cluster). In this case, the user selects a group of clusters and binds the clusters to an arbitrary *binder* cluster. Then, these selected clusters are temporarily bound to the binder cluster instead of their parent clusters until the user nullifies the setting. This option is especially useful to control root clusters (e.g., to make a pony tail style), or to locally control a set of clusters that stem from two disjoint root positions (Figure 4-17). In Figure 4-17a, the cluster A is temporarily attached to cluster B. The cluster A and its descendents are then subject to follow the motion of cluster B. Whenever the user changes the properties of the cluster B, the cluster A is affected by the change as before.

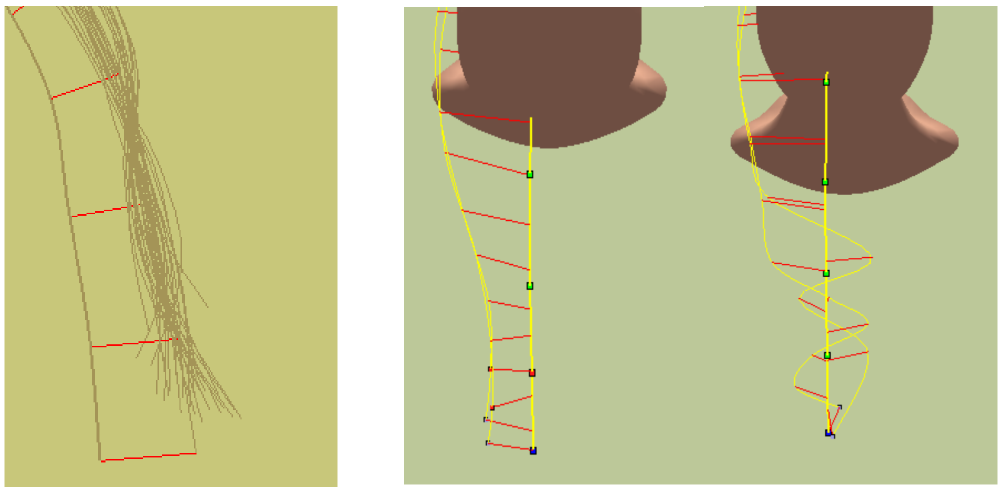

(a) (b)

Figure 4-17. Temporary and local binding. In this figure, only skeleton curves are shown. (a) Temporary binding. (b) Local binding.

In Figure 4-17b, two root clusters are locally bound to another (binder) cluster and only selected control points (red) are twisted around the binder cluster. Note that in these examples, the skeleton curves of the binder cluster are used as the axis of deformation, similarly to the axial deformation techniques such as Wires [Singh 1998].

4.6.2 Copy and Paste

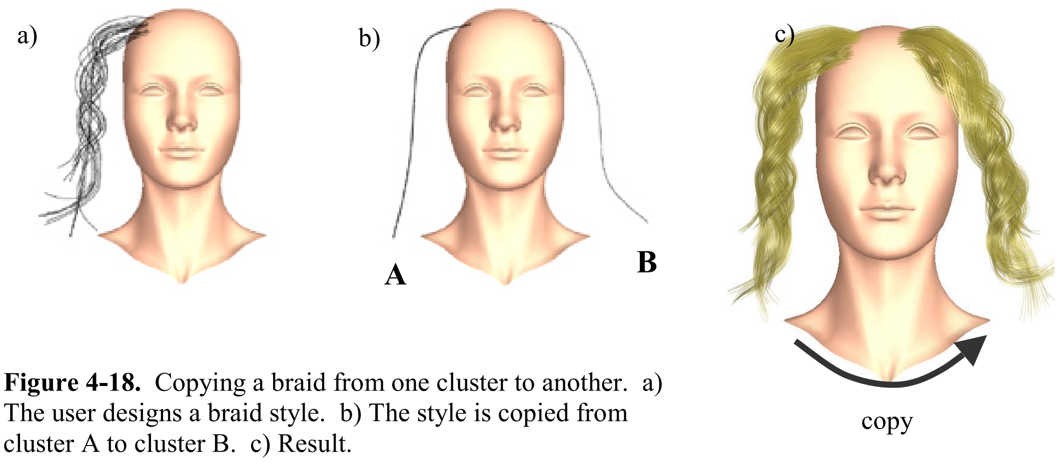


Figure 4-18. Copying a braid from one cluster to another. a) The user designs a braid style. b) The style is copied from cluster A to cluster B. c) Result.

One can treat a hair cluster and its descendent clusters as a user-defined style template. The copy process illustrated in Figure 4-18 transfers a *style* from one cluster to another, by copying the sub-tree parameters. Copying is similar to the editing process in section 4.6.1. When the style of cluster A is copied to cluster B, the descendent clusters of cluster A are first bound to A, but these clusters are updated with cluster B as their parent. If B is a leaf cluster, then new child clusters are created and attached to this cluster in the hair tree. If B is a non-leaf cluster, its descendents are first deleted and these updated clusters replace existing descendents of cluster B. Contours are scaled by the ratio between the size of contours of cluster A and B. Other style parameters such as scale and twist are simply duplicated. Note that any node in the hair tree can be copied to another node regardless of depth; making every edit the user makes a potential style template (Figure 4-19).

By exploiting copy/paste tools, user can first design a rough set of hairstyles and apply different details to the same hairstyles. It is also often useful to constrain the parameters in the copy function. For example, one can copy only the scale (or twist) parameters of one cluster to another, while keeping other parameters unaffected. In examples shown in Figure 4-1 and in section 4.8, these copy/paste functions were extensively used to speed up the modeling process.

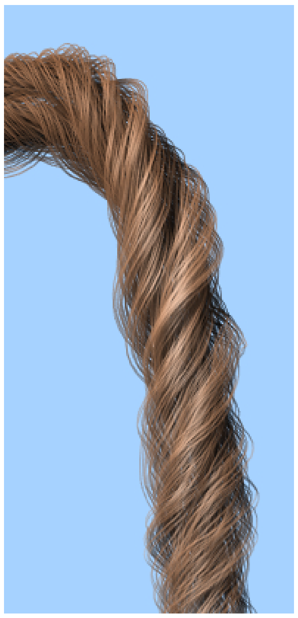


Figure 4-19. A braid style resulting from multiple copy/paste operations at multiple scales.

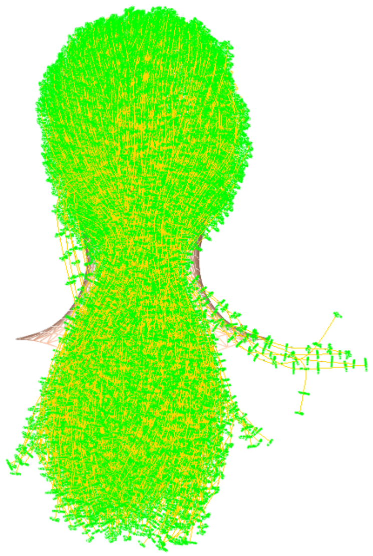


Figure 4-20. The complexity of a multiresolution hair model. Contours (green) and skeleton curves (yellow) of 1340 leaf clusters are shown.

4.6.3 Selection Methods

When a hair model is subdivided into many small clusters, it becomes difficult to select some portions of the model due to its volumetric nature (Figure 4-20). The following selection methods are available. 1) The user picks a control point of a cluster or specifies a region with a sphere. 2) The user selects clusters in scalp space with standard 2D selection methods (for example, dragging a rectangle). 3) The user picks a cluster and traverses the hair tree. Operations such as 'pick first child node' or 'pick next sibling' are mapped to the keyboard, which proves useful in manipulating the multiresolution model. 4) Selection based on depth is provided (e.g., 'display only the clusters of depth < 2').

4.6.4 Avoiding Hair-Head Penetration

It becomes visually distracting if hair strands penetrate the head. During editing, hair strands and skeleton curves are tested for intersection (Figure 4-22). We use a simple iterative algorithm based on the closest-point-transform (CPT) [Mausch 2000]. The CPT provides a fast look-up table solution with the preprocessing of the distances and gradients to the closest points on the head mesh (Figure 4-21). Let D(**p**) be the distance from a point **p** to the closest point on the mesh and $\nabla(\mathbf{p})$ be the gradient of the distance. Penetration is avoided by altering each point **p** inside the head mesh (D(**p**) < 0) with equation (4-9).

$$\mathbf{p}_{\text{new}} = \mathbf{p} - \mathbf{D}(\mathbf{p}) \cdot \nabla(\mathbf{p}) \tag{4-9}$$

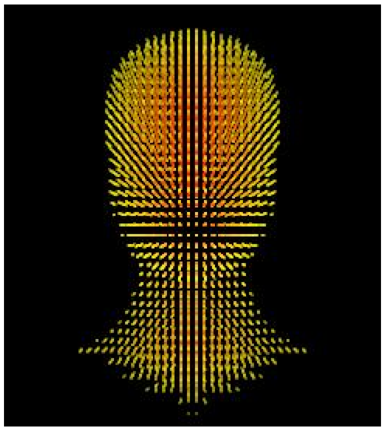


Figure 4-21. Closest point transform of the head mesh. Distance from the mesh is color coded, with red for inner points to yellow for outer ones.

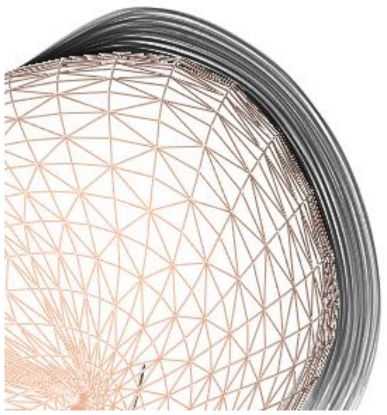


Figure 4-22. Avoiding hair–head penetration.

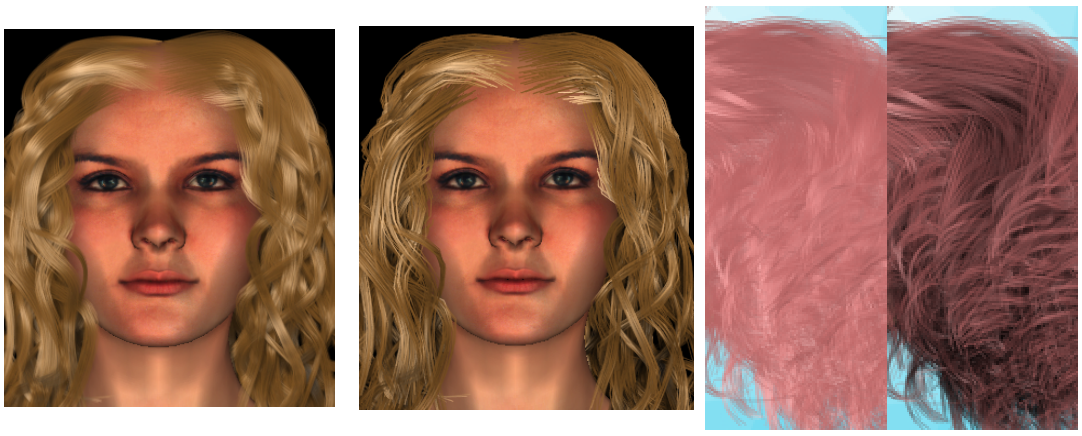
4.7 Level of Detail and Interactive Rendering

Hair models in our framework are explicitly rendered; every hair strand is drawn as polylines in OpenGL. The system renders the edited hair model complete with antialiasing, shadowing, and local shading. See Figure 4-23. With explicit hair models, the results of rendering (e.g., shadows, colors, etc.) can be cached, allowing users to interactively view the model during editing. Users can control three parameters - alpha values, the number of hair strands, and the number of segments per strand -, to adjust the speed of rendering (Figure 4-23a). These parameters give users choices in the tradeoff between speed of rendering and image quality. Increasing the number of hair strands and the number of segments per strand contributes to improved rendering quality, whereas decreasing these parameters allows the user to interactively examine the hair model at higher frame rates. The details of the interactive rendering algorithms are given in chapter 6.

Shading	1.2 second	Visibility Ordering	0.43 second
Hair update	5000 strands per second	Shadow Calculation	5.88 seconds*

Table 4-1. Time measurements for components of MHM system.
*Measurement per light source

The current implementation runs on a system with an Intel PIII 700 Mhz CPU, 512 MB memory, and nVidia Quadro 2 Pro graphics card. Table 4-1 shows time measurements for a hair model of 10,000 hair strands and 40 segments per each strand. For shadow calculation, 40 opacity maps of 200 x 200 pixels were used. During the interactive rendering sessions, a typical setting is to use about 150,000 segments (e.g., 5000 hair strands with 30 segments per strand) with alpha value of 0.5 in a window size of 512 by 512. The system interactively (> 5 frames per second) renders the edited model with antialiasing.



(a) (b)

Figure 4-23. Effects of shadows and level of detail. Images were captured during interactive user sessions. (a) left: 1,200,000 segments (20000 strands, 60 segments per strand, and $\alpha = 0.3$), right: 100,000 segments (5000 strands, 20 segments per strand, and $\alpha = 1.0$). The left model is 12 times more complex than right model, hence the rendering time is 12 times slower for the left model. (b) left: without shadows, right: with shadows.

4.8 Results

Users can create a variety of complex models (Figure 4-24 ~ Figure 4-27). Depending on the complexity, it takes from 10~20 minutes to a few hours to model each hairstyle. The most time-consuming step in modeling is the positioning of initial root clusters. Typically users design 10 to 30 root clusters, which consumes about 70 ~ 80 percent of the total modeling time. Figure 4-24 illustrates an example hair model created with the MHM system.

Hairstylists often use curling and combing as their means to promote clustering effects, adding visual richness due to shadows. Likewise, we observe that adding more hair strands does not necessarily enhance the visual complexity of a hair model. As the hair volume is filled with more strands, room for shadows diminishes. Thus, we find that, to model a natural hairstyle, the *structural* aspects are as important as individual strand details. Note that the spaces between clusters after subdivision amplify shadows between clusters, enhancing the perceived complexity of the model. Shadowing is less salient for smooth hairstyles. However, subdividing smooth hair models can also add visually interesting combing effects (Figure 4-25).

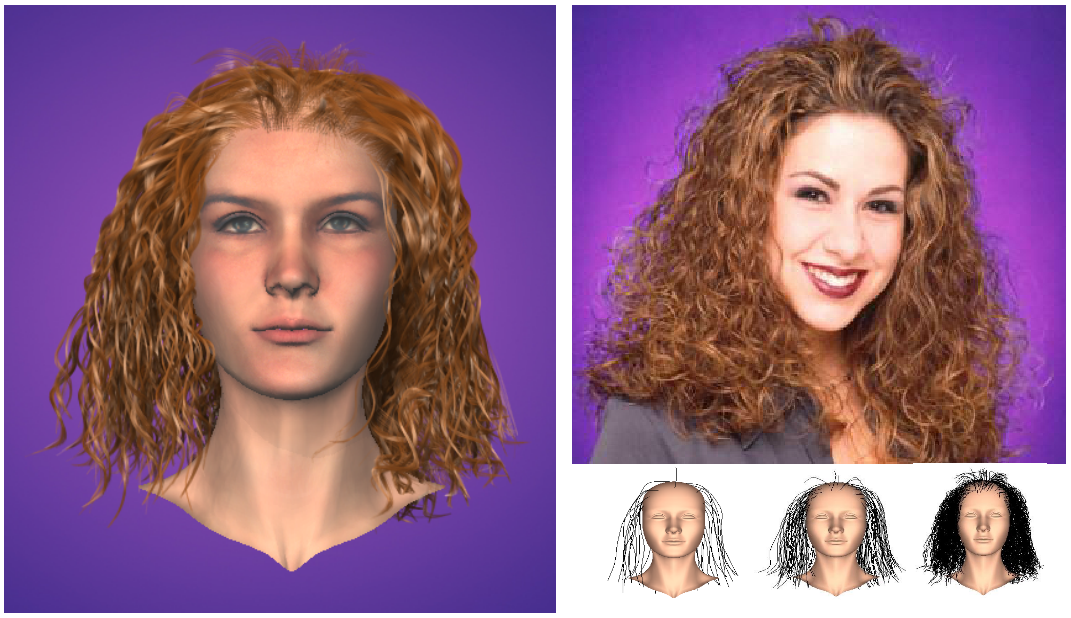


Figure 4-24. Designing a complex hairstyle. The hair model (left) consisting of 940 clusters at three levels of detail was created after the photograph shown in the middle (image courtesy of http://www.hairboutique.com). The small images show the skeleton curves at each level of detail.

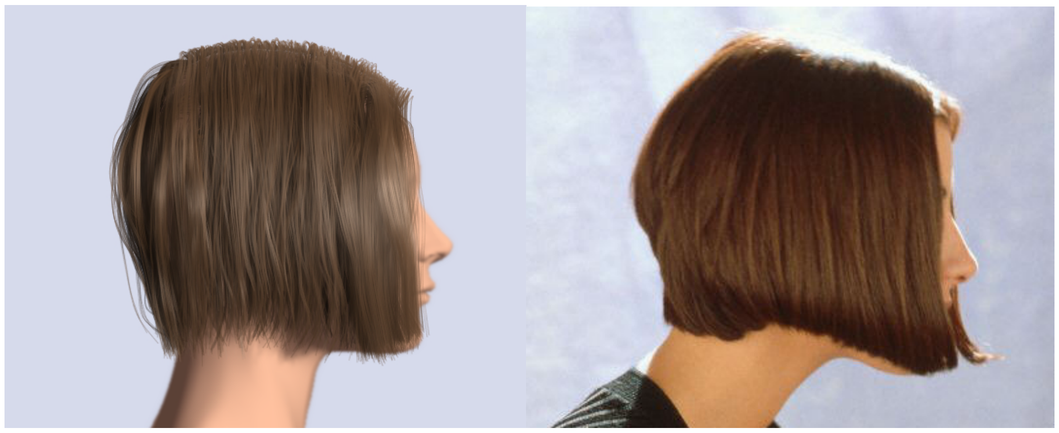


Figure 4-25. A smooth hairstyle. The hairstyle on the left is modeled after the image on the right (http://www.hairboutique.com)

Various hairstyles created with the MHM system are shown in Figure 4-26. Note how the same face model looks different with varying hairstyles [LaFrance 2001]. The strength of MHM system is the user's ability to rapidly edit and add details to existing hair models. For example, the second hair model on the first row was created by editing the final hair model shown in Figure 4-1. Only the shape of root clusters was modified in this case.

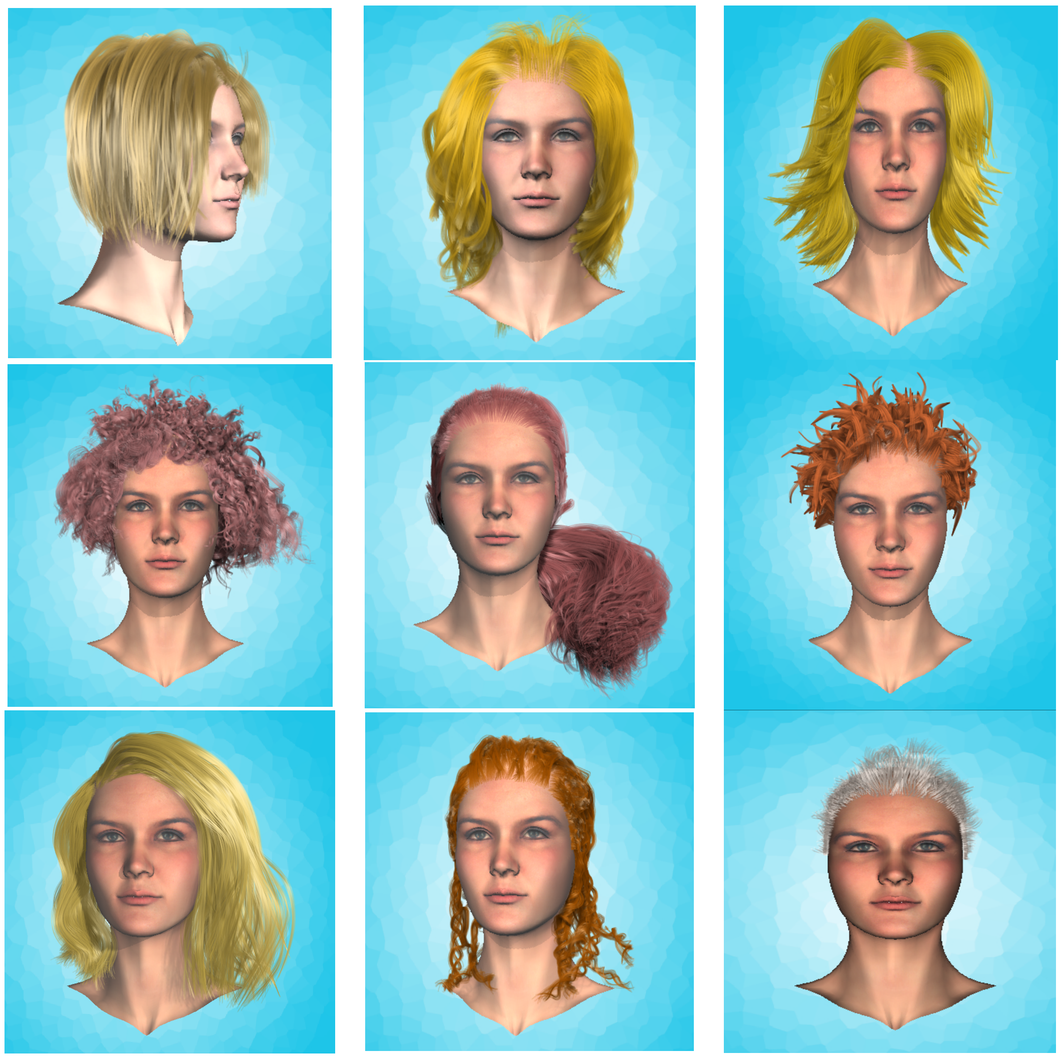


Figure 4-26. More hairstyles.

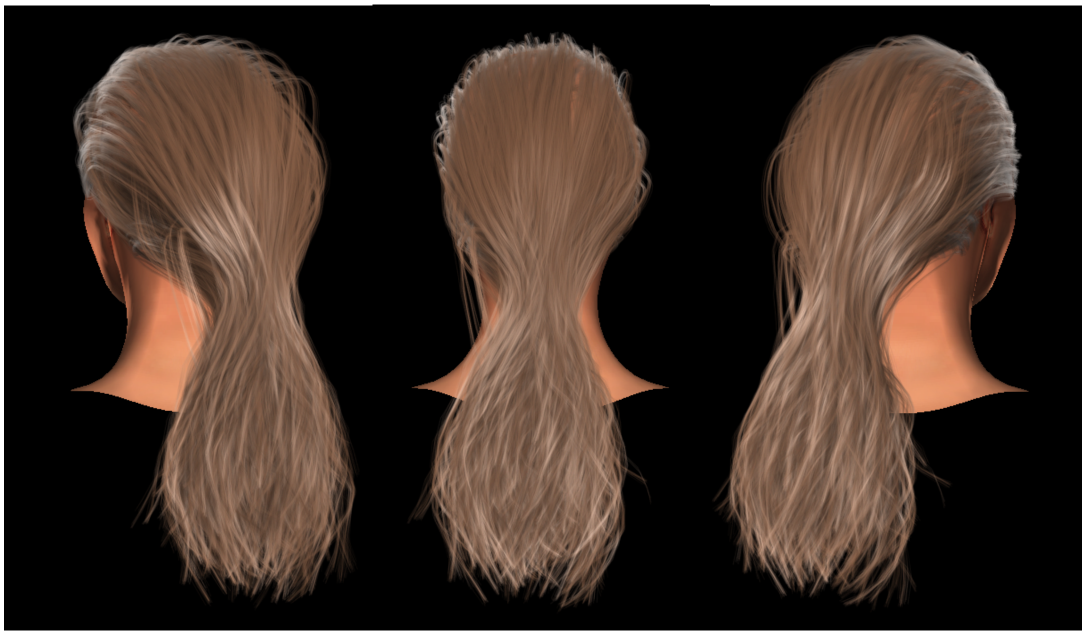


Figure 4-27. Curly ponytail.

4.9 Implementation Details

To speed up the model updates, we use caching schemes such as tabulation of sine functions and curve approximation to find the closest point on a curve. Thus, memory requirement is currently high (200 MB at maximum). Considering that the data structure of the hair tree itself is compact (4KB per each GC), the memory usage could be reduced by further optimizations and faster CPUs. The current bottleneck in the model update is the curve evaluation for each strand (table 4-1). For efficiency, we tag hair clusters that the user is currently editing and update hair strands only for these clusters.

When the root positions of hair strands are sampled, the hair density can vary over the curved scalp surface. Although this is not a serious problem, we could provide an additional density map on the scalp that the user can paint. The scalp surface may also be used as an atlas for texture maps of other parameters such as colors, thickness, etc.

4.10 Conclusion

As with any multiresolution approach, the benefit of MHM is maximized with complex models, such as natural curly hairs. The major benefit of MHM lies in a user's capability to rapidly add structural details and variations, given initial templates. Providing a gallery of rough style templates would greatly speed up the production of various hairstyles.

GCs are chosen as cluster representation mainly due to its modeling efficiency (e.g., axial scaling/twist) and also because a hair strand can be treated as a very thin GC. However, GCs may not be best suited to modeling global shapes for smooth hair or short hair. Existing methods may suffice for smoothly varying hairstyles (e.g., [Anjyo 1992] [Hadap 2000]) and short hair/animal fur (e.g, [Goldman 1997] [Lengyel 2000] [Lengyel 2001]). Thus, it may be worth investigating methods to fit root clusters into other control primitives (e.g., polygonal models) or using MHM as a *backend* system to refine the results from other modeling methods.

Currently, the hair model is defined for a specific head model. It would be useful to transfer hairstyles from one head model to another. The scalp surface abstraction and the copy/paste tool may provide good starting points. Scattered data interpolation techniques such as Radial Basis Functions may also provide reasonable adaptation of the hair model to different head meshes. Completely automating the process may require sophisticated dynamics to handle both hair-hair and hair-head interactions.

In the MHM framework, the user implicitly *designs* the hair/hair interactions in the form of multiresolution clusters. Extending MHM to support animation seems feasible in many ways. At the highest level, one could provide the user with kinematics or dynamics controls over root clusters. Alternatively, strand level animation techniques could be employed to animate the lowest level strands. However, in reality when hair moves, hair/hair interactions cause hair clusters to change dynamically, from large coherently moving clusters to independently moving strands, and vice versa. Simulation of such dynamic clustering effects is a challenging problem. The preservation of user-defined style raises another issue. See chapter 7 for more discussions on animation.

This chapter concludes the series of hair modeling work I have done during the course of my Ph.D. study. As in the TSV method presented in the previous chapter, efforts were made to reproduce the visual structure of human hairstyles, with the help of combing functions in the TSV model or multiresolution cluster model in this chapter. A hair model is not a simple unorganized soup of hair strands. There are always structures in the form of clustering, curling, and other hair-hair interaction effects. These structures are what hairstylists often strive to create through their extensive styling efforts. These volumetric structures form a visual signature that differentiates one hairstyle from another. The differences in visual appearance of hair not only comes from the shape differences, but also comes from the way hairs reflect, scatter, and occlude the illumination around them. The next two chapter present detailed theoretical foundation and practical algorithms related to the problem of reproducing the visual appearance – hair rendering.

Chapter 5

5. Fundamentals of Hair Image Synthesis

Hair rendering is challenging in many aspects of computer graphics. First of all, the number complexity causes the image synthesis process to be computationally demanding. For example, in a recent movie 'final fantasy', more than 25% of rendering time is spent purely for hair rendering [Duncan 2001].

Hair, as a collection, is a highly discontinuous volume. The volumetric nature of hair is not simply captured by any surface based rendering algorithms. The hair volume is partly translucent, as observed in back lighting situations where dominant lights are placed behind the hair. The discontinuous and thin geometry of hair models cause serious under-sampling problems known as aliasing in computer graphics. A naïvely implemented hair rendering algorithm often causes visually objectionable high frequency images, failing to capture the smooth appearance of hair. The reflectance property of each hair strand is not yet well understood, adding to the difficulty of hair rendering.

Although there exist some impressive hair-rendering work from the film industry, few details have been published, with exceptions such as [Berney 2000] [Bruderlin 1999] [Goldmann 1997] [Fong 2001]. A framework is thus needed to understand the fundamental problem and to provide a guideline for any research on hair rendering. This chapter aims at providing an overview of the general problem of hair rendering

In general, one needs to take into account the followings to realistically render hair,

- *Anisotropy (reflectance model)*; hairs reflect and scatter lights differently depending on the light's direction and the viewing direction in a complicated way.
- *Global illumination*; the scattering and inter-reflection between hairs create important visual cues that distinguish one hairstyle from others. Different illumination conditions due to indirect light transport result in rich color variation even for hairs consisting of similar material properties. In particular, self-shadows (attenuated light) provide the sense of depth and volume for the underlying structure.
- *Antialiasing*; the geometry to be rendered is highly discontinuous. Hence the correct sampling of the thin geometry is an essential step in realistic hair rendering.

There are a few fundamental theories that govern any image synthesis (rendering) algorithms. Section 5.1 introduces some important concepts that were developed in the field of signal processing. Then, the general equations of volume rendering are described in the context of hair rendering in section 5.2 through section 5.6.

5.1 Aliasing and Sampling Theory

Alasing is a well-studied problem with a long history, first identified by Crow in computer graphics [Crow 1977]. The sampling theorem, developed in the signal processing community, states that the sampling frequency of continuous signal should be at least double the highest frequency of sampled signal [Shannon 1949]. This minimum sampling frequency is often referred to as the Nyquist frequency. In the context of hair rendering, the theorem can be restated as follows. The sampling frequency of image samples should be higher than double the highest frequency of underlying hair model. Since a hair fiber has a highly asymmetric shape with extremely small cross sections, the highest variation in hair geometry mostly comes from the thickness of a hair strand.

It is known that a hair fiber is as thin as 0.03 to 0.09mm [Valkovic 1988]. In a situation where the hair volume covers the viewer's field of view, a hair strand covers only 0.045 to 0.135 degrees of the field of view, assuming that the viewer's field of view is about 30 degree and each dimension of the hair volume is about 20 cm long. The projected size of a hair strand is equivalent to the size of a single pixel in an image of 2000 to 7000 pixels along each dimension. The Nyquist frequency for hair rendering becomes about 0.0225 to 0.07 degrees, or 15 to 40 cycles per degree, or 4000 to 14000 pixel resolution per each dimension of the image. As the viewer moves farther from the object (hair), the required sampling frequency turns even higher. Moreover, when many hair strands partially occlude each other, the required sampling frequency can potentially go even higher than the simple estimation above. In a common environment setting, the image resolution rarely reaches such dimension. In an image device with the resolution of 640 by 480 pixels, it is common that dozens or even hundreds of strands contribute to the color of a single pixel.

Two most widely used techniques for rendering are perhaps the ray tracing [Glasnner 1995] [Whitted 1980] and Z-buffer based rasterization methods [Foley 1995]. Both methods heavily rely on point sampling; each pixel is considered an infinitesimal point sample of the scene. The point sampling corresponds to the ray-object intersection test in ray tracing and the center of scan conversion grid in Z-buffer based rasterization. For point sampling to work correctly, the signal (color, shadow, reflectance) coming from the object should be first band limited [Carlson 1986]. In other words, the signal should not vary inside the pixel's extent. However, as already noted, each pixel can be covered by many hair strands and a single point sample cannot capture this variance. For example, see Figure 5-1. Assume that three hair strands cover a pixel and each hair strand's color is red, green, and blue, respectively. If each hair strand covers exactly one third of the pixel's extent, the correct color sample of the pixel should be an averaged color of the three colors - gray. However, a single point sampling will cause the pixel to change in color to either of the three. So, instead of gray (a true sample), the pixel's color will alternate in red, blue, and green, depending on the point sample's position.

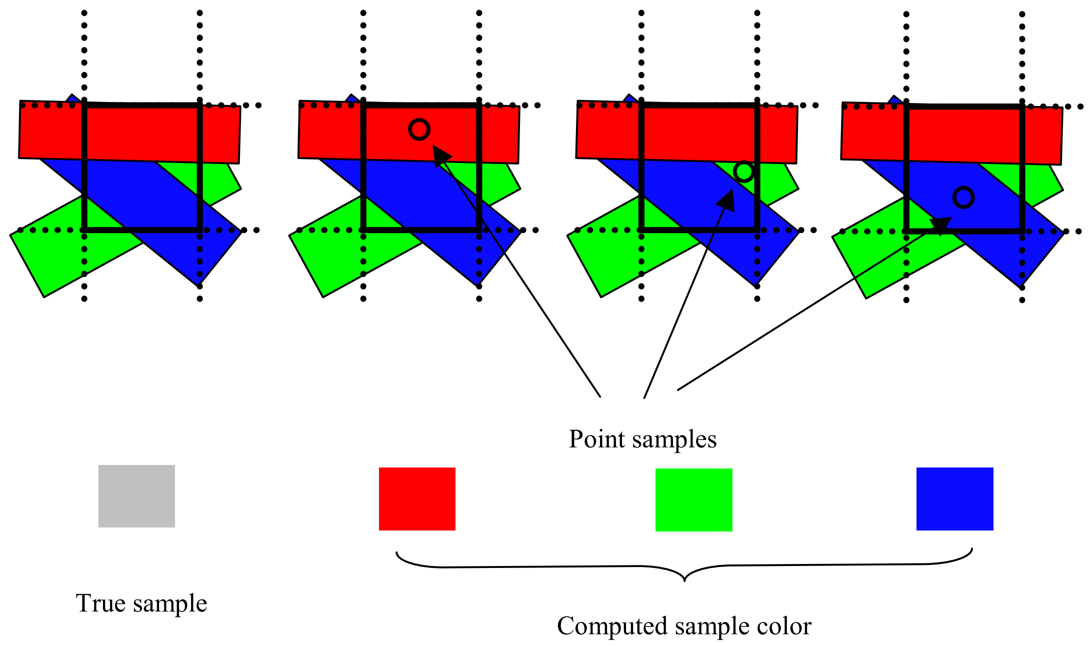


Figure 5-1. Consequences of point sampling in the presence of complex geometry.

This type of undersampling is the primary source of the visually distracting aliasing artifacts such as sparkling of colors and brightness in hair rendering. Supersampling is a natural solution to the alasing problem. The main idea is to use sufficient number of point samples per each pixel, and average their contributions, at the cost of increased time to generate these point samples. It is known from the central limit theorem that increasing the number of samples can reduce the variance. A set of random samples placed on the pixel's extent will exhibit the variance of $O(N^{-1})$ and the standard deviation of $O(N^{-1/2})$ [Mitchel 1996]. This implies that as more samples are used, the sample will converge to a true integral over the pixel's extent, at the rate of $O(N^{-1/2})$. The Nyquist threshold should be reached for convergence. However, as discussed earlier, the Nyquist frequency is very high for hair rendering, and the convergence rate is slow. The standard deviation of samples is a rough measure to errors in samples [Mitchell 1996]. This error measure only increases by $O(N^{-1/2})$. For example, using 100 samples are only 3.3 times better than 10 samples in terms of this error metric. So, in practice, for any super sampling method to work well for hair, a significant number of samples should be used (and it is a common practice in industry). In the context of rendering algorithm, a ray tracer needs hundreds or thousands of samples per pixel and hardware rasterization would need hundreds or thousands of accumulation steps.

The stratified sampling techniques, where sub-pixel grids are used to produce jittered sample patterns, often increase the convergence rate of the error for some scenes [Mitchell 1996]. However, the techniques are known to be no better than random samples in the case of highly complex geometry such as hair [Mitchell 1996] [Lokovic 2000]. In terms of signal processing, both random samples and stratified samples tend to push errors toward high frequency areas that are visually less distracting in the case of smooth geometry. However, as discussed in chapter 2, a digital image of hair has a broad energy spectrum that often spans over high frequency region. Thus, even stratified sampling or uncorrelated jittering [Keller 2001] is not much helpful in the case of hair rendering.

An efficient way of distributing point samples [Dippe 1985] [Mitchell 1987] [Mitchell 1996] [Keller 2001], or different ways of sampling [Jones 2000] continue to be an active area of research. As mentioned earlier, reasonable reconstruction of a high frequency signal can be only achieved by band limiting the original high frequency signal. In this case, the reconstructed signal represents a low-pass filtering of the original signal. Band limiting requires integration of the scene's contributions over a pixel's sampling extent (or sampling kernel). In the context of supersampling a large number of samples should be computed and then averaged per each pixel, as current rendering algorithms and imaging devices hardly reach the Nyquist limit for hair. A good hair rendering algorithm thus should implement a low pass filter for the underlying high frequency signals.

Starting from the next section, the remainder of this chapter will discuss the properties of the actual samples that are needed for hair rendering such as radiance, shadows, colors.

5.2 Volume Rendering Equation for Hair Rendering

Rendering a digital image of hair involves estimation of the radiance observed by a viewer for any point *x* inside the hair volume. The general equations for rendering a volume can be stated as [Kajiya 1984; Jensen 2001a],

$$\begin{aligned} L(x,\vec{\omega}) &= \int_0^s e^{-\tau(x,x')}\sigma_a(x')L_e(x')dx' \\ &+ \int_0^s e^{-\tau(x,x')}\sigma_s(x')\int_{\Omega_{4\pi}} p(x',\vec{\omega}',\vec{\omega})L_i(x',\vec{\omega}')d\vec{\omega}'dx' \\ &+ e^{-\tau(x,x+s\vec{\omega})}L(x+s\vec{\omega},\vec{\omega}) \end{aligned} \tag{5-1}$$

The equation (5-1) states the amount of radiance going out from the point x along a path of length s in direction of ω. The first term describes the sum of emitted light along the path and the second term accounts for the radiance from other side of the volume due to scattering, and the third term describes the attenuated propagation of light from the end point of the path. The transmittance function τ, (or optical depth [Jensen 2001a]), is a line integral of densities along a linear path.

$$\tau(x,x') = \int_x^{x'} \sigma_t(t)dt \tag{5-2}$$

Three coefficients σ_a ,σ_s, σ_t govern the relative amount of light absorbed, scattered, and cancelled (due to both absorption and scattering), respectively.

Since hair does not emit any light, the first term can be ignored if all the lights are assumed to be outside the hair volume. The volume rendering equation may be more intuitively interpreted as follows. Consider the second term of the equation (5-1) that involves nested integrals over a path (outer integral) and over all the directions (inner integral).

$$\int_0^s e^{-\tau(x,x')}\sigma_s(x')\int_{\Omega_{4\pi}} p(x',\vec{\omega}',\vec{\omega})L_i(x',\vec{\omega}')d\vec{\omega}'dx' \tag{5-3}$$

The inner integral term states that for the point *x*, another point *x'* along the path scatters the light it receives from all the possible directions (ω'), toward the direction of the path (ω). If a lighting function I is defined as

$$I(x',\omega) = \int_{\Omega_{4\pi}} p(x',\vec{\omega}',\vec{\omega}) L_i(x',\vec{\omega}') d\vec{\omega}' \qquad (5\text{-}4)$$

, the volume rendering equation can be then compactly rewritten as

$$L(x,\vec{\omega}) = \int_0^s e^{-\tau(x,x')} \sigma_s(x') I(x',\vec{\omega}) dx' + e^{-\tau(x,x+s\vec{\omega})} L(x+s\vec{\omega},\vec{\omega}) \qquad (5\text{-}5)$$

This equation governs the process of hair rendering. The first term tells that the amount of radiance that a point x receives in direction ω is an attenuated integral of radiance that are in-scattered from other points. The second term conversely indicates that the radiance at the other point along the path will also contribute to the radiance at point x in a similar way. Aside from the scattering coefficient σ_s, the most important terms in hair rendering are the phase function and the exponential attenuation. The phase function dictates the way a light is *locally* reflected, scattered, and transmitted through thousands or even millions of hair fibers. The exponential attenuation term accounts for a global variation of radiance due to varying densities of hair fibers inside the hair volume. The direct measurement of this attenuation forms a dominant factor in self-shadows between hair fibers.

5.3 The Phase Function

The reflectance and scattering of light inside a hair fiber can be stated as a three-dimensional function analogous to the BRDF (Bidirectional Reflectance Distribution Function) functions that are widely used for surface rendering. It is important to note that a hair reflectance model (phase function) should not only compute the backward scattering (reflectance such as BRDF), but also the forward scattering (transmission). This phase function is often referred to as a local shading model in hair rendering literature. In these shading models, only the information about local neighborhood of a point in consideration is used.

A phase function for hair rendering can be stated as five dimensional function of the position on a hair fiber, and the incoming and outgoing directions of radiance.

$$\int_{\Omega_{4\pi}} p(x, \vec{\omega}', \vec{\omega}) d\vec{\omega}' = 1 \tag{5-6}$$

This function tells the ratio of the incoming radiance in direction of ω and the outgoing radiance in direction of ω’ on the position *x* on a hair fiber. Note that the phase function is similar to the well-known BRDF function, but this function is unitless and normalized and a full sphere of directions are used instead of a hemisphere in this case. If the computation is limited to the local neighborhood of the position, however, this function can be reduced to four dimensions, consisting only of the incoming and outgoing directions transformed to the local coordinate system on the hair fiber (Figure 5-2).

This reduction of dimension assumes that the function does not vary along a hair fiber’s length. Furthermore, assuming that the cross section of a hair fiber is roughly radially symmetric (close to a circle)[9], another dimension can be reduced. An arbitrary normal vector N can be used

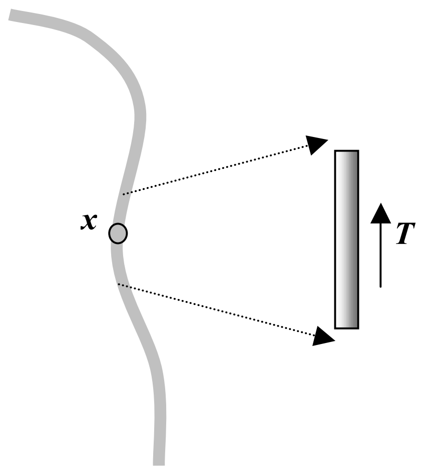


Figure 5-2. A hair strand is assumed to be a straight cylinder in the neighborhood of the point **x** for shading calculation. The differential cylinder has a tangent vector **T** around **x**.

[9] It is known that the cross section of a hair fiber is slightly elliptic [Valkovic 1988]. However, if we consider the number of hair fibers, the orientations of these cross sections are random enough to assume that the cross section is a circle.

such that the normal is perpendicular to the tangent direction and both the normal and the incoming direction ω stay on a plane perpendicular to the tangent direction (Figure 5-3). Thus, a local coordinate frame can be constructed around the point x, with the normal vector and the tangent vector. Given this frame, the phase function becomes three dimensional, where only the zenith angle θ is meaningful for the incoming direction ω (**L** in Figure 5-3).

The notion of phase function [Kajiya 1984] [Jensen 2001a] is widely used in the literature of astrophysics. Commonly used phase functions include perfect diffuse scattering function, Rayleigh scattering [Kajiya 1984], and the Henyey-Greenstein phase function [Henyey 1947]. See [Jensen 2001a] for examples. The phase functions in these contexts are fairly simple and often isotropic (independent of directions). The phase functions are based on an assumption that the participating

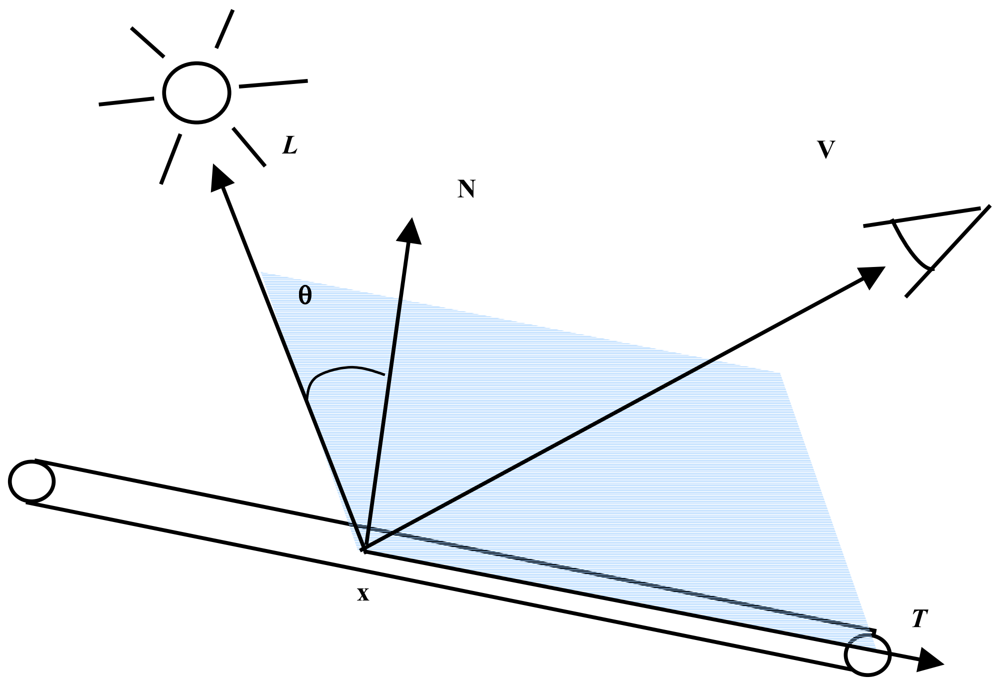


Figure 5-3. Hair shading geometry

scattering particles are spherically shaped. Since scattering from a sphere is isotropic, only the phase angle, the angle between the incoming direction and the outgoing direction is used. In the case of hair fiber, however, the shape is rather cylindrical and highly anisotropic. A specialized phase function is thus needed for hair rendering.

Because of the anisotropy, the direction of hair growth represented by the tangent vector of the cylinder becomes the dominant factor for hair shading. A normal vector, often a dominant factor for surface shading models such as Phong shading and BRDFs, is not well defined for hair shading. As mentioned in the previous section, it is often assumed that the optical behavior of a hair fiber is independent of the orientation around the tangent vector. In this case, one can choose any normal vector on the plane perpendicular to the tangent vector (Figure 5-4). In practice, it is convenient to define a local coordinate frame around the cylinder. A widely used scheme is to define the normal vector as the projection of the light direction on the normal plane [Anjyo 1992] [Banks 1994] [Kajiya 1989].

With this local frame coordinate system, a hair shading function can be represented as $p(\theta, \theta' \phi')$, a phase function of three variables θ, the angle between the light direction **L** and the normal **N** of local frame coordinate system, φ and θ', the polar coordinates of the reflection vector

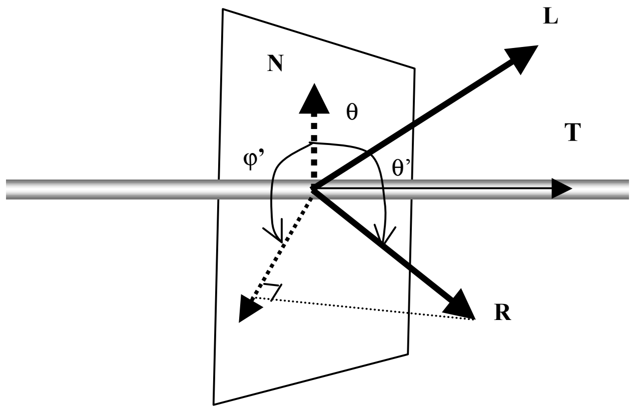


Figure 5-4. The reflection vector R can be expressed by polar coordinates of the local coordinate frame defined by the tangent and the normal plane.

R in the local coordinate frame (Figure 5-4). Thus, hair shading models mentioned above can be considered to compute this three-dimensional phase function $p(\theta, \theta' \phi')$ in one form or another.

5.4 Hair Shading Model

A shading model states how much a hair fiber reflects or transmits to a given direction when the hair fiber is fully lit. Since global aspects such as shadows are not accounted for, the term *local shading model* is often used. It is often assumed that the shape of a hair strand on its local neighborhood is a straight cylinder (Figure 5-2).

The most commonly used shading model is the one originally developed by Kajiya and Kay [Kajiya 1989]. The model is composed of Lambertian diffuse component and anistropic specular component. Similar, but simpler models are used in [Anjyo 1992] and [Rossenblum 1991].

In the Kajiya-Kay model, a Lambertian cosine falloff function is used for diffuse lighting. The closer the light is to the normal, the more illumination is received.

$$\Psi_{Diffuse} = K_d \sin(T, L) \tag{5-7}$$

, where K_d is a scaling constant for the diffuse illumination and (V_1,V_2) denotes the angle between two vectors V_1,V_2. In [Anjyo 1992], the same quantity is computed using the angle between $\vec{L}$ and $\vec{N}$, or θ.

$$\Psi_{Diffuse} = K_d \sin(T, L) = K_d \cos(N, L) = K_d \cos\theta = K_d (N \cdot L) \tag{5-8}$$

A non-diffuse (specular) illumination is computed using the viewing vector $\vec{V}$**.** The specular illumination becomes the biggest when the angles between $\vec{L}$**,** $\vec{N}$ and $\vec{N}$**,** $\vec{V}$ are the same. The Kajiya-Kay model computes

$$\Psi_{Specular} = K_s [(T \cdot L)(T \cdot V) + \sin(T, L)\sin(T, V)]^p \tag{5-9}$$

Using the polar coordinates, Equation (5-9) can be compactly rewritten as

$$\begin{aligned}\Psi_{Specular} &= K_s[(T\cdot L)(T\cdot V)+\sin(T,L)\sin(T,V)]^p \\ &= K_s[\cos(T,L)\cos(T,V)+\sin(T,L)\sin(T,V)]^p \\ &= K_s\cos^p[(T,L)-(T,V)] \\ &= K_s\cos^p[(N,L)-(N,V)] \\ &= K_s\cos^p(\theta-\theta')\end{aligned} \tag{5-10}$$

Putting the diffuse and specular terms together, the Kajiya-Kay model becomes

$$p(\theta,\theta',\phi') = K_a + K_d\cos(\theta) + K_s\cos^p(\theta-\theta') \tag{5-11}$$

Note that the shading model is independent of the azimuth angle *φ'* and it depends only on the relative zenith angle. This shading model is simple, efficient and thus widely used. However, a real hair fiber is more complicated than this simple model. As pointed out in [Goldman 1997], a hair fiber is not perfectly diffuse around its cylindrical axis. As introduced in chapter 2, a hair fiber is layered and semi-transparent. Due to scattering, a hair fiber exhibits strong transmission (forward scattering) when lit from behind, while it is more diffuse when lit from front.

Based on the above observation, Goldman adds a scalar term, *κ*, that computes the directionality of a hair fiber with regard to the lighting direction and the view direction.

$$\kappa = \cos(T\times L, T\times V) = \frac{(T\times L)\cdot(T\times V)}{\|T\times L\|\cdot\|T\times V\|} \tag{5-12}$$

The directional attenuation factor is then given as

$$f_{dir} = \frac{1+\kappa}{2}\rho_{reflect} + \frac{1-\kappa}{2}\rho_{transmission} \tag{5-13}$$

If the reflection from the underlying head surface is ignored, the Goldman model can be written as

$$\Psi_{hair} = f_{dir}(\Psi_{Diffuse} + \Psi_{Specular}) \tag{5-14}$$

Note that the directional term is a function of three vectors **T**, **L**, and **V**. If we align the y-axis of the local coordinate frame with **N**, and the x-axis with **T**,

$$\begin{array}{lcccc} T & = & [1 & 0 & 0]^T \\ L & = & [\sin(\theta) & \cos(\theta) & 0]^T \\ V & = & [\sin(\theta')\cos(\varphi') & \cos(\theta') & \sin(\theta')\sin(\varphi')]^T \end{array} \tag{5-15}$$

The two cross products in Equation (5-12) are then,

$$\begin{array}{lcccc} T\times L & = & [0 & 0 & \cos(\theta)]^T \\ T\times V & = & [0 & -\sin(\theta')\sin(\varphi') & \cos(\theta')]^T \end{array} \tag{5-16}$$

Plugging these terms into Equation (5-12) yields

$$\begin{aligned} \kappa &= \cos(T\times L, T\times V) = \frac{(T\times L)\cdot(T\times V)}{\|T\times L\|\cdot\|T\times V\|} \\ &= [0 \quad 0 \quad 1]^T \cdot [0 \quad -\sin(\theta')\sin(\varphi') \quad \cos(\theta')]^T / \|T\times V\| \\ &= \frac{\cos(\theta')}{\sqrt{\sin^2(\theta')\sin^2(\varphi') + \cos^2(\theta')}} \end{aligned} \tag{5-17}$$

and

$$\begin{aligned} f_{dir} &= \frac{\sqrt{\sin^2(\theta')\sin^2(\varphi')+\cos^2(\theta')} + \cos(\theta')}{2\sqrt{\sin^2(\theta')\sin^2(\varphi')+\cos^2(\theta')}}\rho_{reflect} \\ &+ \frac{\sqrt{\sin^2(\theta')\sin^2(\varphi')+\cos^2(\theta')} - \cos(\theta')}{2\sqrt{\sin^2(\theta')\sin^2(\varphi')+\cos^2(\theta')}}\rho_{transmit} \end{aligned} \tag{5-18}$$

Note that the directionality term κ and hence the attenuation term f_{dir} are the function of two variables θ',φ'. The directionality term κ becomes positive if $\|\theta'\| < \frac{\pi}{2}$, and negative if $\|\theta'\| > \frac{\pi}{2}$. The positive case corresponds to the frontlighting when the light and view direction are at the same side of the hair, and the negative case corresponds to the backlighting. Note also that two additional control parameters are added in this model, ρ_{reflect} and ρ_{transmit}. These parameters are empirically set. For example, nearly equal coefficients are used for white or gray hairs and higher reflectance value is used for pigmented hair [Goldman 1997]. Given equally set coefficients, the attenuation coefficient f_{dir} simply becomes a constant. In this case, the Goldman model is equivalent to the Kajiya-Kay model.

These existing models are purely empirical. The parameters such as diffuse, specular, and directionality coefficients should be controlled at the user's discretion. One obvious problem may be that these shading models completely ignore the multiple reflections inside the fiber. As a consequence, the diffuse terms become zero when the lighting direction coincides with the tengent vector. In reality, non-trivial amount of lights can be observed even in such case. These result in parameter tweaking for practical use in the film industry [Berney 2000].

5.5 Yet Another Shading Model

In a search for better shading models, I experimented with two extended versions of the previous shading models. One is purely empirical and another is based on simplified physical simulation of hair fiber. This subsection will briefly describe these shading models. In particular, the shading models are inspired by the observation that the scattering patterns of a fiber are significantly different depending on whether the light is in the same direction as the observer (front lighting) or not (back lighting).

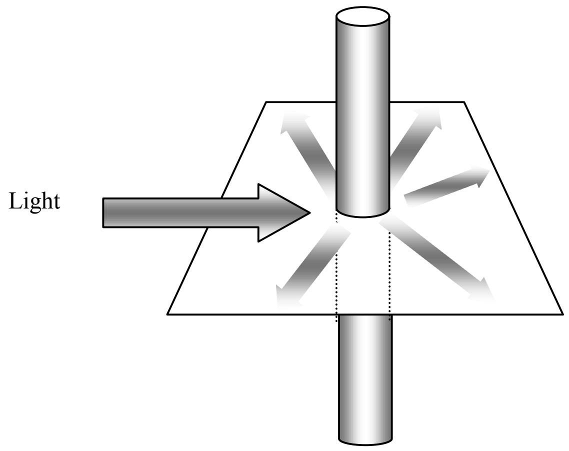


Figure 5-5. A cross-section of a semi-transparent cylinder illuminated by a directional light.

In this subsection, discussions are limited to a situation where the light is hitting a hair fiber in a perpendicular direction to the fiber. In other word, assume that the light vector lies on the normal plane (Figure 5-5). Recall that the phase function $p(\theta,\theta',\phi')$ has three arguments. Assume that the phase function is separable as follows.

$$p(\theta,\theta',\phi') = p'(\theta,\theta')I(\phi') \tag{5-19}$$

The main focus here is to examine how the angle (ϕ') around the cylindrical axis of a hair fiber affects the phase function.

5.5.1 Emprical Model

Consider the cross section of a cylinder by a plane perpendicular to the tangent (Figure 5-6). Given a directional light, Lambertian illumination incident on the cross sectional circle is given as

$$I(\phi') = \begin{cases} I\cos(\phi'), -\frac{\pi}{2} \le \phi' \le \frac{\pi}{2} \\ 0, otherwise \end{cases} \tag{5-20}$$

,where I is incident illumination, and ϕ' is the azimuth angle between the light's direction $\vec{L}$ and a vector $\vec{C}$ connecting the center and a point on the circle (Figure 5-6a).

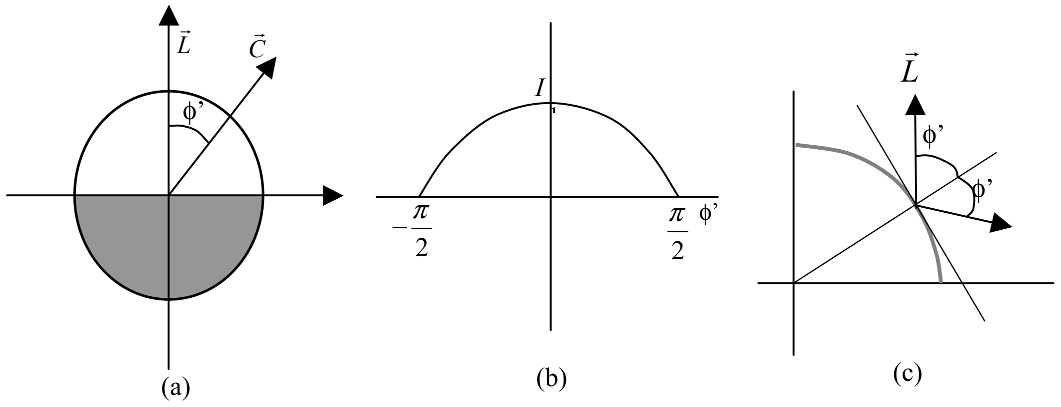
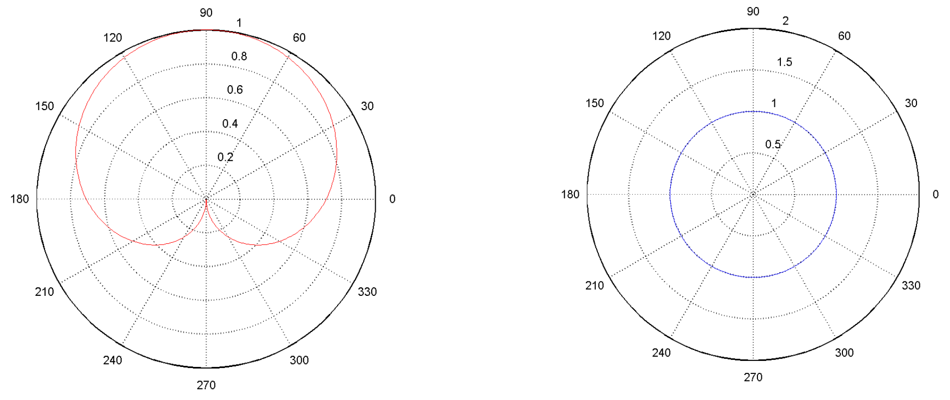


Figure 5-6. Light reflected off a circle. (a) The gray region is not illuminated by light in direction of $\vec{L}$. (b) Lambertian illumination as a function of θ. (c) Perfect mirror reflection of incident ray.

Assuming that a hair fiber is composed of a homogeneous material and the surface of a hair strand is perfectly smooth, the incident radiance is reflected using a perfect mirror reflection. Given the incident angle ϕ, the angle of reflected ray is 2ϕ (Figure 5-6c). Plugging it into (5-20) results in the following equation.

$$R(\phi') = \begin{cases} I\rho_r \cos(\frac{\phi'}{2}), -\pi \le \phi' \le \pi \\ 0, otherwise \end{cases} \tag{5-21}$$

The polar plot of reflected energy is shaped like a flipped heart (Figure 5-7a). Note that the shape is different from shapes of the widely used Kajiya-Kay model (Figure 5-7b). Light direction is assumed to lie along 90 degree.



(a) Perfect mirror reflection (b) *Kajiya-Kay* model

Figure 5-7. Energy plot of phase functions.

Due to refraction, a hair fiber shows strong forward scattering property and hence the energy plot of transmitted lights tends to be much narrower than in backward scattering (Figure 5-8). The transmitted light can be approximated using a single cosine lobe.

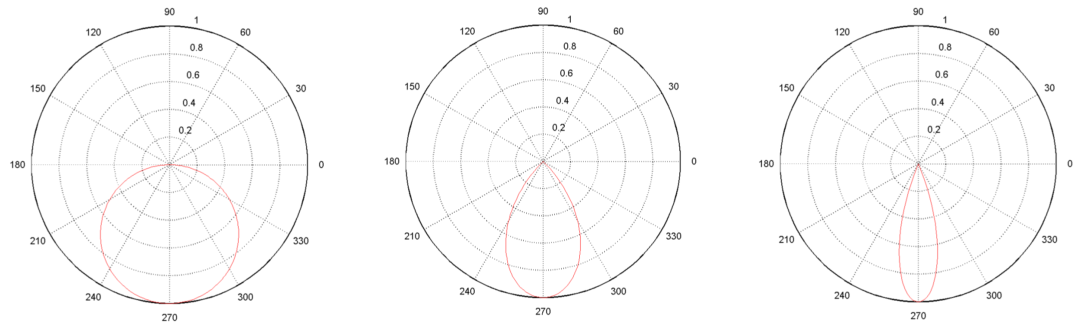
$$T(\phi') = \begin{cases} I\rho_t \cos(k(\phi' - \pi)), |\phi'| \ge \pi\left(1 - \frac{1}{2k}\right) \\ 0, otherwise \end{cases} \tag{5-22}$$

A constant *k(*≥1.0*)* is used to control the amount of refraction such that larger *k* results in narrower transmission.

Combining two phase-functions yields in a single phase equation.

$$I(\phi') = \rho_r R(\phi') + \rho_t T(\phi') \quad . (5\text{-}23)$$

, where ρ_r and ρ_t are the albedo terms to control the amount of scattering. Figure 5-9 illustrates some examples of combined phase functions.



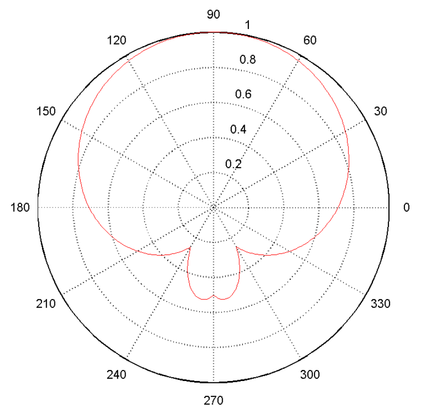
(a) $k = 1.0$ (b) $k = 2.0$ (c) $k = 4.0$

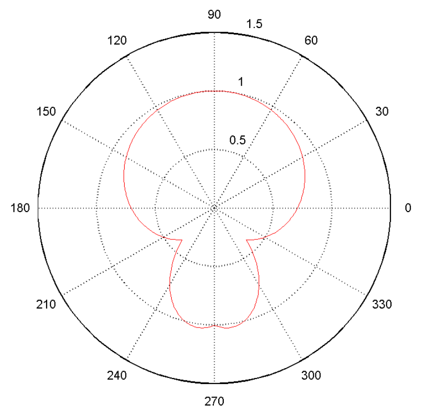
Figure 5.8. Phase functions for forward scattering. *k* controls the sharpness of scattering.

Finally, the empirical model combines above one-dimensional phase function with Kajiya-Kay model (equation 5-11) as

$$p(\theta,\theta',\phi') = p'(\theta,\theta')I(\phi') = \{K_d \cos(\theta) + K_s \cos^p(\theta - \theta')\}\{\rho_r R(\phi') + \rho_t T(\phi')\} \quad (5\text{–}24)$$

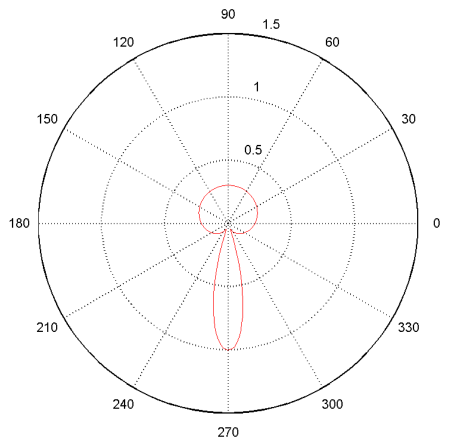
Note that this type of factoring is possible since Kajiya-Kay model depends on the two zenith angles and our phase term depends on only the outgoing azimuth angle. In fact, our model is similar in spirit to the model by Goldmann (equation 5-14). In particular the f_{dir} term is comparable to our phase term (equation 5-23). However, the Goldmann's directionality term is arbitartily chosen and its physical meanings are not very clear (note the complexity of the attenuation term in equation 5-18 when it is written with polar coordinates). Also, the ability to tune the size of each cosine lobe for either reflection or transmission is missing in the Goldman's model.

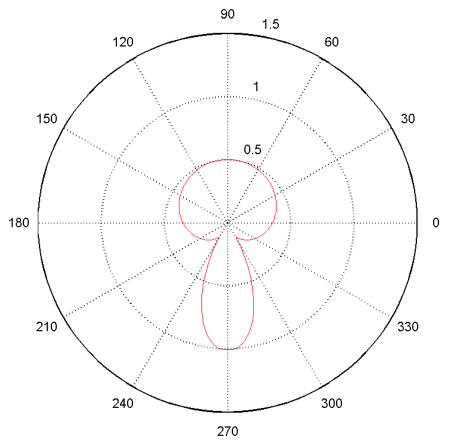




(a) ρ_r=1.0, ρ_t =0.5, k=3.0

(b) ρ_r=1.0, ρ_t =1.0, k=2.0





(c) ρ_r=0.3, ρ_t =1.0, k=5.0

(d) ρ_r=0.5 , ρ_t =1.0, k=3.0

Figure 5-9. Examples of combined phase functions.

5.5.2 Simulated Model with a Single Layer

It is known that a dielectric material such as glass changes its reflectance depending on the angle of incident lights. A glass with the index of refraction 1.50 transmits 96 percent of incoming lights at normal incidence, while fully reflects the incoming light at grazing angle [Jenkins 1976]. Reflectance increases with the angle of incidence, first gradually and then more rapidly. The fraction of reflected lights on transparent material can be expressed using Fresnel's laws of reflection (Figure 5-10).

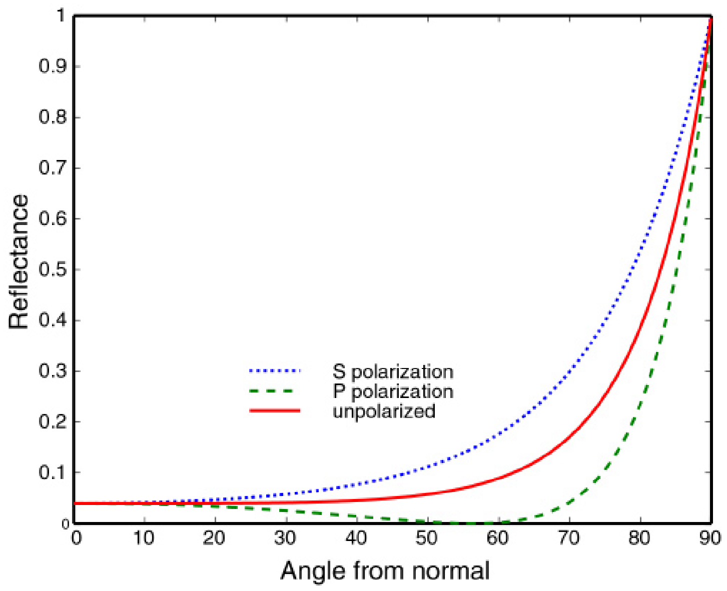
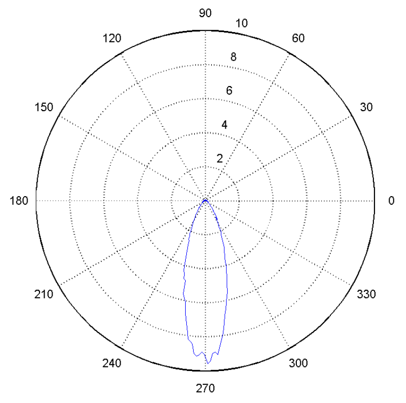


Figure 5-10. Fresnel reflectance for a glass having n = 1.50.

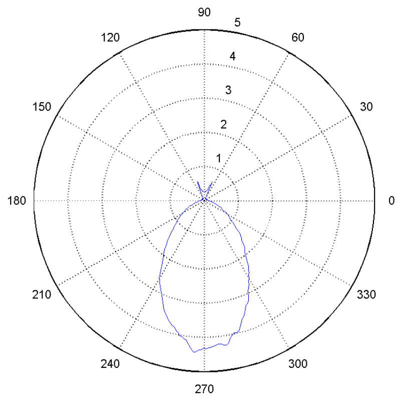
In this experiment, the scattering function I(ϕ') is simulated using a Monte-Carlo sampling technique. Since the reflectance is isotropic around the cross section, the light direction is fixed along the vertical axis. For each ray, the first incident point is sampled on the hemi-circle, and propagated using the following rules.

- The fresnel reflectance is used as a probability of a ray being reflected or refracted.
- Refracted ray is computed using the Snell's law.
- When the ray exits the boundary, the energy is recorded according to the direction.

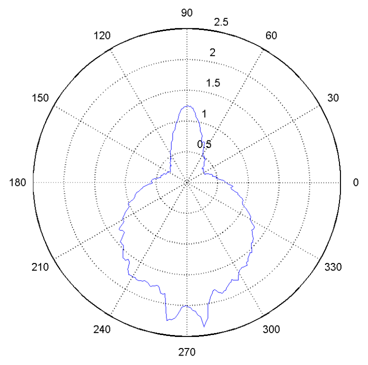
The simulation depends only on the index of refraction. Figure 5-11 shows examples of simulated phase functions. From the optics research [Wang 1995], it is known that the index of refraction for natural hair fibers is about 1.58. In practice, the functions around this value will be more meaningful (Figure 5–11b). Interestingly, it can be seen that the transmission term is much stronger than the reflection term. Also, two small peaks can be observed diagonally in the reflection case.



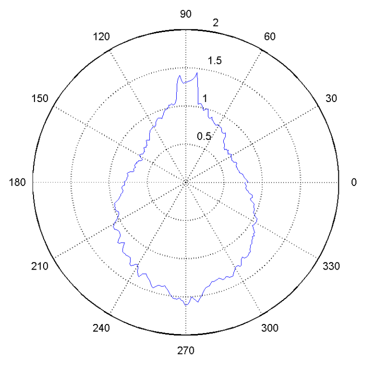
(a) Index of refraction = 1.2



(b) Index of refraction = 1.5

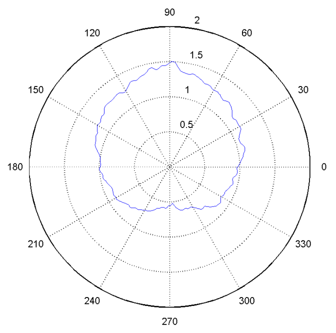


(c) Index of refraction = 2.5

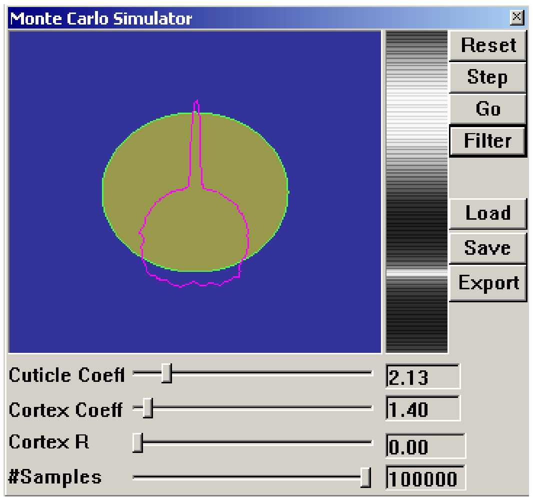


(d) Index of refraction = 3.5

Figure 5-11. Simulated phase functions (continued on the next page).



(e) Index of refraction = 10.0



Cuticle Coeff 2.13
Cortex Coeff 1.40
Cortex R 0.00
#Samples 100000

Snapshot of simulation tool

Figure 5-11. Simulated phase functions (continued from the previous page).

5.5.3 Discussion

The assumption in section 5.5.1 that a hair fiber has a very strong forward scattering property can be verified to a certain extent with this hypothetical experiment. The simple experiment may explain why the primary scattering along the light direction (self-shadows) is the dominant factor in hair rendering. Real human hair fibers are composed of multiple layers (Figure 5-12). A cross section of hair strand reveals two distinct components, the *cuticle* and the *cortex*. The cuticle is the translucent outermost surface of the hair shaft, whilst the inner bulk of the hair is composed of a more fiberous keratin (cortex). To provide a more meaningful simulation result, the simulation of multiple scattering between these two layers may be needed. In addition, full 3D simulation of a hair cylinder may be required instead of a simple 2D model presented here. In this aspect, physically based measurement of the optical properties of human hair fiber can be useful. This is still an active research area in the optics community [Wang 1995]. For physically based rendering of different hair fibers, it may be useful to acquire and store a database of such physical measurement samples, either through tabulation of this three dimensional function, or by higher order fitting techniques such as spherical harmonic functions [Westin 1992].

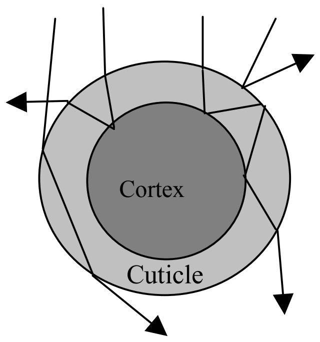


Figure 5-12. A ray can follow many kinds of paths inside a hair fiber.

5.5.4 Recap

In this section, it was shown that any hair shading model can be expressed in the form of the phase function $p(\theta,\theta',\phi')$ in the context of a volume rendering equation. As mentioned above, a traditional hair shading model use three vectors - the light direction $\vec{L}$, hair tangent $\vec{T}$, and the viewing vector $\vec{V}$. The conversion between these vectors and three phase angles can be done as follows.

The normal $\vec{N}$ is defined as in section 5.4. Then two azimuth angles θ, θ' are given as $\theta = \cos^{-1}(\vec{N}\cdot\vec{L})$ and $\theta' = \cos^{-1}(\vec{N}\cdot\vec{V})$. The viewing vector $\vec{V}$ is projected onto the plane perpendicular to $\vec{T}$. The projected vector $\vec{v}$ is given as $\vec{v} = \vec{V} - (\vec{V}\cdot\vec{T})\vec{T}$. The phase angle ϕ' is then given as $\phi' = \cos^{-1}(\overrightarrow{N}\cdot\vec{v})$. Then these angles can be used to compute hair reflectance with any hair shading model described so far.

5.6 Shadows and Primary Scattering

Recall the volume rendering equation (5.5), restated below

$$L(x,\vec{\omega}) = \int_0^s e^{-\tau(x,x')}\sigma_s(x')I(x',\vec{\omega})dx' + e^{-\tau(x,x+s\vec{\omega})}L(x+s\vec{\omega},\vec{\omega})$$

The volume rendering equation can be further reduced to a simpler form if multiple scattering is ignored. The assumption made here is that the light marches and attenuates along a straight path and get scattered only once before it is sensed by the viewer. This is equivalent to approximating the inner integral of the volume rendering equation with the direct illumination

$$\int_{\Omega_{4\pi}} p(x',\vec{\omega}',\vec{\omega})L_i(x',\vec{\omega}')d\vec{\omega}' = L(x',\vec{\omega}) = L(x+s\vec{\omega},\vec{\omega}) \qquad (5\text{-}25)$$

In other words, the in-scattered light can be considered being convoluted with a dirac filter with the preferred direction toward the light. Let $\vec{\omega}_l$ be such a direction. Then, the volume rendering equation reduces to

$$\frac{L(x,\vec{\omega}_l)}{L(x+s\vec{\omega}_l,\vec{\omega}_l)} = \int_0^s e^{-\tau(x,x')}\sigma_s(x')dx' + e^{-\tau(x,x+s\vec{\omega}_l)} \qquad (5\text{-}26)$$

Since scattering is ignored the first term can be discarded. Here $L(x,\vec{\omega}_l)$ is the amount of attenuated (shadowed) direct illumination that a point *x'* receives from the light. Rearranging the equation yields

$$L(x,\vec{\omega}_l) = e^{-\tau(x,x+s\vec{\omega}_l)}L(x+s\vec{\omega}_l,\vec{\omega}_l) \qquad (5\text{-}27)$$

Equation (5-27) is a simplified volume rendering equation that accounts for only forward scattering. This equation will be approximated through an efficient shadow algorithm in chapter 6. Combining this with the primary scattering yields a simplified radiance estimate for each viewing direction $\vec{\omega}$ as

$$L(x,\vec{\omega}) = \sum_l p(x,\vec{\omega},\vec{\omega}_l)e^{-\tau(x,x+s\vec{\omega}_l)}I_l \qquad (5\text{-}28)$$

I_l is a constant illumination from the light source *l*. Equation (5-28) states that the attenuated contribution from each light source is scattered due to the phase function and these contributions are summed over each light. Note that the first term computes the summation of radiation from the light sources through the dense hair volume. Note that this is essentially equivalent to existing hair rendering methods that combine shadow maps with local shading models. This observation forms the theoretical basis for the efficient hair shadow and antialiasing algorithms that will be presented in the chapter 6. There are many simplifying assumptions with it and these should be noted if more accurate physical simulations are desired.

5.7 Sampling the Radiance

Let ψ (u,v) be the radiance sample observed at the point on the image plane parameterized by (u,v). Then, a correct sample for each pixel should be computed as

$$I_{i,j} = \int_{-r}^{r} \int_{-r}^{r} f(s,t)\psi(i+1/2-s, j+1/2-t)dsdt \tag{5-29}$$

, *f* is a weighting kernel function and ψ is the line integral of radiance defined above.

In practice, it is important to note that the accumulated radiance estimate is affected by two dominating factors. First, fractional visibility should be correctly computed as a hair strand rarely fully covers a pixel's extent. This is accounted for in equation (5-29) with the weighting kernel. Second, a hair strand is not a completely opaque object. The correct treatment of such translucency is important to faithfully capture the optical properties of hair, regardless of the number of samples used for each pixel.

5.8 Other Issues

Previous sections discussed equations that are fundamental for hair rendering. The volume rendering equation for hair describes, per each pixel sample, the accumulated measure for radiance. Although direct conversion of radiance estimate to RGB colors often yields a reasonably good digital picture of hair, there are issues that demand more cares.

- High dynamic range imaging: It is known that visual response of human eyes is a nonlinear function of the actual radiance [Durand 2000]. A corresponding conversion operator, often referred to as tone mapping, may be needed for certain kinds of scenes where the dynamic range of measured radiance is substantially large as in the backlighting in outdoors.

- Color responses: It is also known that the commonly used RGB color space does not form a perceptually consistent linear mapping of colors. Simply scaling a RGB color does not produce the same tone of color. This is important for hair rendering where color of hair strands continuously vary due to self-shadows and varying illumination. Thus, simply scaling a yellow color does not reproduce the feel of a blond hair color. A normalized color space such as XYZ space [Glassner 1995] could be a better option.

5.9 Hardware Accelerated Hair Rendering

The enormous number of hair strands necessitates a computationally efficient rendering algorithm. However, the underlying fundamentals of image synthesis introduced in this chapter reveal a number of difficulties that make a physically correct simulation of light transport inside a hair volume very computationally demanding and practically impossible. Instead an alternate formula for practical rendering was formulated. In the next chapter, a practical hair rendering system is introduced based on the simplified rendering model. The goal of such modification and simplification is to build a hair rendering system that is efficient enough to be used in an interactive hair modeling framework, but is still faithful to the underlying principles of hair image synthesis described in this chapter. More specifically, the next chapter will target for developing scalable and efficient hair rendering algorithms for explicit hair models that may vary in time.

Ray tracing is the most general method to any kind of lighting simulation. However, as discussed earlier, ray tracing requires a large number of samples per pixel and the calculation of each sample could be prohibitively expensive, considering the large number of hair strands at hand. Moreover, correct simulation of translucency adds to the complexity.

The work presented in the next chapter is motivated by the recent progress in computer graphics hardware. Considering the numeric complexity of an explicit hair model (often over a million segments), it seems natural to exploit the ever-increasing performance of graphics hardware as much as possible. However, the standard graphics pipelines are not designed for complex, thin geometry such as hair. Nevertheless, there exist some ways to exploit this valuable resource for hair rendering. In the next chapter, a set of efficient hair rendering algorithms are developed, each of which heavily exploits existing graphics hardware. This leads to interactive rendering of arbitrary explicit hair models, complete with shading, antialiasing, and self-shadows.

Chapter 6

6. Interactive Hair Rendering

This chapter presents a practical interactive hair rendering system tightly integrated in the interactive hair modeling system in chapter 4. Since a hair volume can form virtual any shape through hairstyling or in animation, it is necessary for a rendering system to handle arbitrary explicit hair model. The discussions in this chapter are limited to explicit hair models since they are the most general hair representation. Example hair models are from chapter 4, but models from any existing hair modeling system could be also used, as long as they are explicitly represented.

6.1 Overview

With an explicit hair model, each hair strand is drawn separately. From a rendering system's point of view, a hair strand can be represented as polygonal strips, or cylinder, or a parametric curve. Since a hair strand is typically much thinner than the size of a pixel, these two representations are in fact equivalent inside graphics hardware (a line is treated as the special case of thin polygon). In the rendering system presented here, each hair strand is represented by a set of line segments (polyline). Figure 6-1 illustrates the components of the system. Given a face model and a hair model, the system produces an image in roughly three steps – computing shadows and colors of each hair (polyline), drawing the face model, and compositing colored line drawing of hairs.

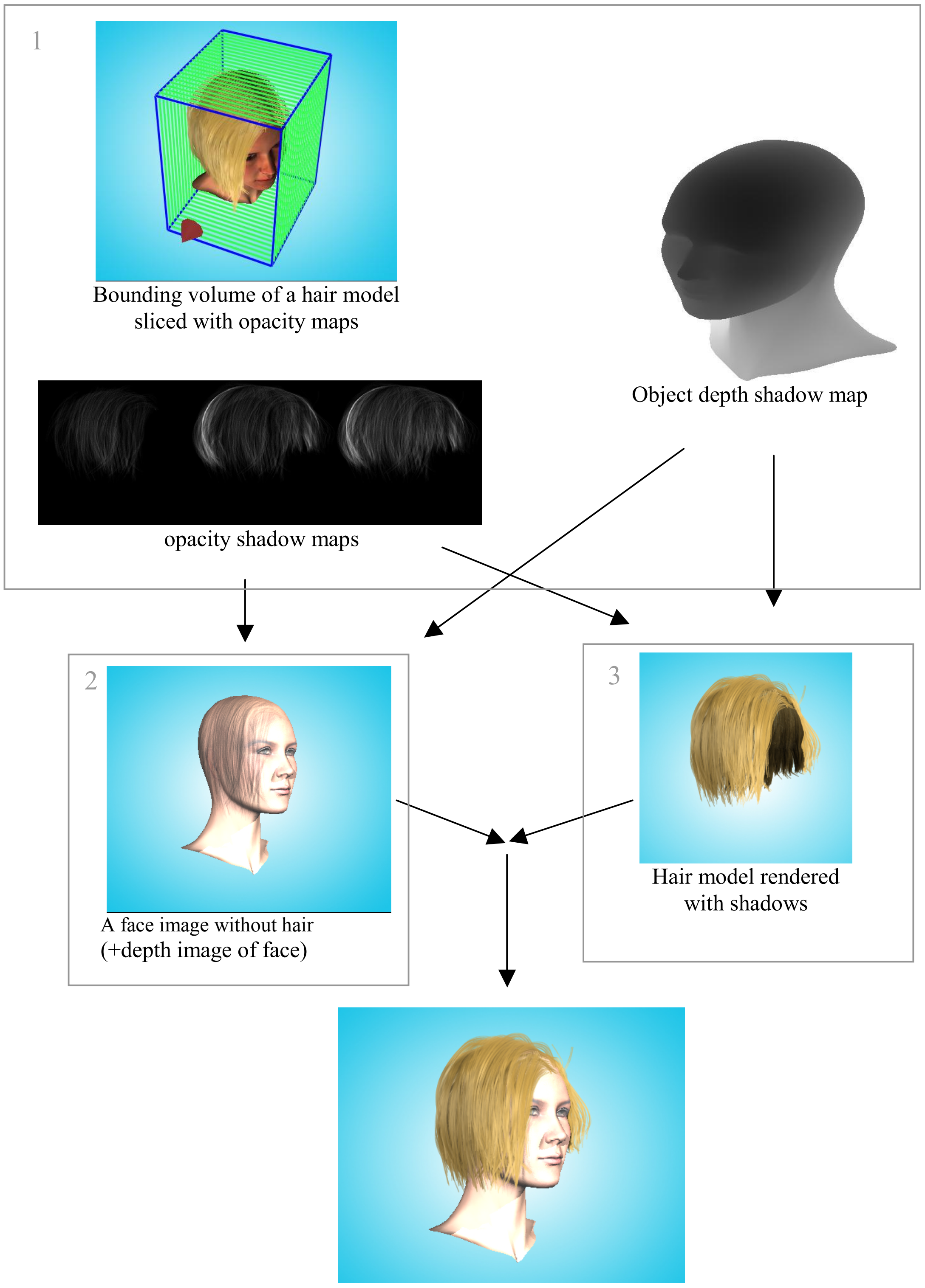

1
Bounding volume of a hair model sliced with opacity maps
Object depth shadow map
opacity shadow maps
2
A face image without hair (+depth image of face)
3
Hair model rendered with shadows

Figure 6-1. Overview of the hair rendering process

Below are brief descriptions of each component. In later sections, each component will be discussed in more details.

1. Shadows are computed at each end point of the line segment. Hair self-shadows and shadows from hair to other objects are computed with the opacity shadow maps algorithm (section 6.2). Shadows from other objects to hair or object self-shadows are computed with the depth-based shadow maps. Since shadows are view-independent, a static hair model of modest complexity can be interactively rendered by caching pre-computed shadow values for each hair strand.
2. There are two options to integrate other scene objects (e.g. face). For interactive viewing purpose, other objects are first drawn with OpenGL and the depth buffer update is disabled. Finally, line segments representing hairs are then drawn. In this case, shadows from hair to objects and object self-shadows are not computed. Alternatively, the scene objects can be pre-rendered with an offline renderer. With an offline render, an image of other objects with a depth map is precomputed. First the image is drawn into the OpenGL color buffer and the depth map is also loaded to the OpenGL Z-buffer. Then, the depth buffer update is disabled and hairs are rendered with the same mechanism as above.
3. Shading calculation is performed at the end points of each line segment (Kajiya-Kay model is used). The shading calculation is done for each light source and multiplied by the transmittance value representing shadows. These color values are summed over all the light sources. The computed color between those endpoints are interpolated by simply drawing each line with the two colors in OpenGL (analogous to Gouraud shading for polygon rendering). Lighting options in OpenGL are disabled. For correctly filtered line drawing, antialiasing is performed in two passes (section 6.3). Line segments are first ordered by visibility. In final drawing pass, the ordered lines are drawn from farthest to closest to the viewer.

6.2 Hair Self-shadows by Opacity Maps*

Hairs cast shadows onto each other, as well as receive and cast shadows from/to other objects in the scene. Especially, self-shadows create crucial patterns that distinguish one hairstyle from others. Brute force self-shadow calculation may result in an $O(N^2)$ algorithm. Conventional depth-based shadow map algorithms incur severe shadow aliasing problems. This section introduces an efficient shadow generation algorithm that makes full use of graphics hardware accelerator.

Rendering self-shadows inside volumetric objects (hair, fur, smoke, and cloud) is a challenging problem. Unlike solid objects, a dense volume made of many small particles exhibits complex light propagation patterns. Each particle 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 generation difficult. However, self-shadows are crucial to capture effects such as backlighting.

Two techniques are generally used for volumetric shadows[10]; shadow maps [Lokovic 2000] [Reeves 1987] [Williams 1978] and ray tracing [Glassner 1989] [Kajiya 1984] [Jensen 2001a]. In traditional depth-based shadow maps (DBSM), the scene is rendered from the light's point of view and the depth values are stored. Each point to be shadowed is projected to the light's camera and the point's depth is checked against the depth in the shadow map. In ray tracing, a ray is shot from a scene point to the light. If the ray intersects any particle on its way, shadows are detected and accumulated. Complete ray tracing of hairs can be prohibitive in terms of rendering time. In practice, shadow maps are often used in conjunction with ray tracing for efficiency.

A good property of traditional shadow map is that the shadow map can be generated with hardware by rendering the scene from the light's point of view and storing the resulting depth buffer. However, severe aliasing artifacts can occur with small semi-transparent objects. As

* Presented in Eurographics Rendering Workshop, London, 2001.

[10] For other types of shadows, refer to [Woo 1990] and survey sections in [Zhang 1998] [Lokovic 2000] for more recent ones.

discussed in section 5.1, in a dense volume made of small primitives, depths can vary radically over small changes in image space. The discrete nature of depth sampling limits DBSM in handling such objects. Moreover, small particles are often semi-transparent due to forward scattering. The binary decision in depth testing inherently precludes such translucency. Thus, DBSM is unsuited for volumetric objects such as hairs.

The transmittance τ(*p*) [Lokovic 2000] of a light to a point *p* can be written as

$$\tau(p) = \exp(-\Omega)\text{, where } \Omega = \int_0^l \sigma_t(l')dl' \qquad (6\text{-}1)$$

In (6-1), l is the length of a path from the light to the point, σ_t is the extinction (or a density) function along the path, as given in equation (5-2) and [Blinn1982] [Kajiya1984] [Lokovic2000]. Ω is the *opacity* value at the point. Note that the term opacity is often used to denote the actual shadow. Here the term *opacity* is used for brevity to denote the accumulative extinction function. More precise term for Ω will be *opacity thickness* as described in [Pattanaik 1993]. Note also that equation (6-1) is a simplified version of volume rendering equation given in equation (5-27).

In the deep shadow maps (DSM) [Lokovic2000], each pixel stores a piecewise linear approximation of the transmittance function instead of a single depth, yielding more precise shadow computation than DBSM. The transmittance function accounts for two important properties of hair.

- **Partial visibility:** In the context of shadow maps, the transmittance function can be viewed as a partial visibility function from the light's point of view. If more hairs are seen along the path from the light, the light will be more attenuated (occluded), resulting in less illumination (shadow). As noted earlier (recall Figure 5-1), visibility can change drastically over the pixel's extent. The transmittance function handles this partial visibility problem by correctly integrating and filtering all the contributions from the underlying geometry.
- **Translucency :**As noted in section 5.3 and section 5.4, 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.

Despite the compactness and quality, however, DSM requires a significant amount of data initialization time. Due to the underlying data structure (linked list), hardware acceleration becomes difficult. When the volume changes in time with regard to the light (e.g. hair animation or interactive hair modeling), the generation cost can cancel out the computational benefit of the algorithm.

Opacity shadow maps (OSM) provide an alternative way to compute the transmittance function. Opacity shadow maps algorithm uses a set of parallel opacity maps oriented perpendicular to the light's direction (Figure 6-2). By approximating the transmittance function with discrete planar maps, opacity maps can be efficiently generated with hardware. On each opacity map, the scene is rendered from the light's point of view, clipped by the map's depth (Figure 6-3). Instead of storing depth values, each pixel stores Ω, the line integral of densities along the path from the light to the pixel. The opacity values from adjacent maps are then sampled and interpolated during rendering.

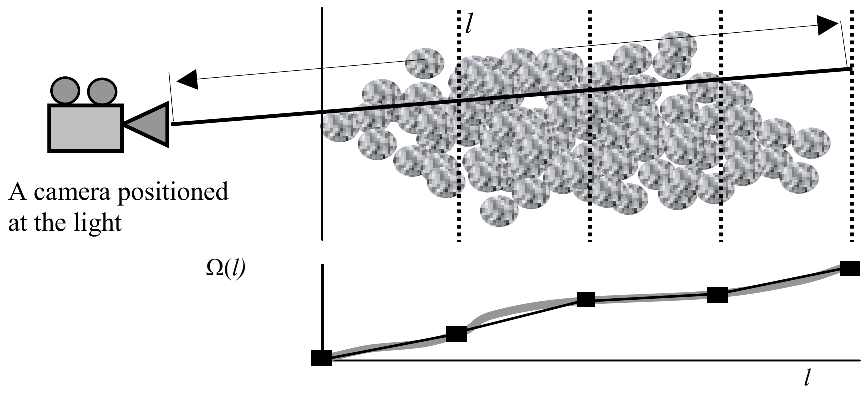


Figure 6-2. The opacity function $\Omega(l)$ shown in a solid gray curve is approximated by a set of opacity maps.

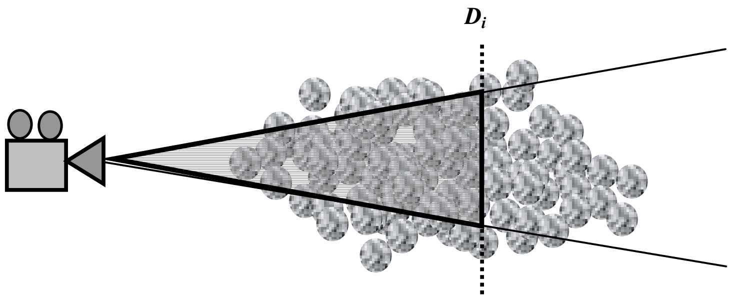


Figure 6-3. The volume is rendered on each opacity map, clipped by the map's depth (***D**$_i$*). The transparent region illustrates the clipped volume.

Although OSM resembles volume rendering [Drebin 1988] [Levoy 1989], a major distinction is that OSM uses a volume of explicit geometry primitives (points, lines, and polygons) instead of a sampled scalar-field (voxels). Thus, it does not incur high memory requirements for a full voxel data array. Also, OSM maintains the object space accuracy of the scene model unlike volume rendering where precision is subject to sampling density.

The idea of opacity maps was exploited in feature films. For example, in the movie *'Mission to Mars'*, a software renderer was used to render the sand-vortex sequence using an SGI origin machine with 16 processors [Lewis 2001]. However, these attempts do not consider hardware acceleration and hence the rendering took a substantial amount of time (about an hour per frame).

6.2.1 Basic Algorithm

Opacity shadow maps heavily rely on graphics hardware and operate on any bounded volumes represented by standard primitives such as points, lines and polygons. (In our context, hairs are represented as a cluster of lines.) The hair volume is sliced with a set of opacity map planes perpendicular to the light's direction. The scene is rendered to the alpha buffer, clipped by each map's depth. Each primitive contributes its associated alpha value.[11] Each pixel in the map stores an alpha value that approximates the opacity relative to the light at the pixel's position. The

[11] The alpha value is a user-controllable parameter that depends on the size (thickness) and the optical property (albedo) of hair. It also depends on the resolution of the opacity maps.

opacity values of adjacent maps are sampled and linearly interpolated at the position of each shadow computation point, to be used in a shadowed shading calculation.

The pseudo code in table 6-1 uses the following notation. P is the set of all the shadow computation sample points (or simply *shadow samples*). N is the number of maps and M is the number of shadow samples. D_i is the distance from the opacity map plane to the light ($1 \leq i \leq N$). P_i is a set of shadow samples that reside between D_i and D_{i-1}. p_j is j_{th} shadow sample ($1 \leq j \leq M$). *Depth*(p) returns a distance from p to the light. $\Omega(p_j)$ stores the opacity at p_j. $\tau(p_j)$ is the transmittance at p_j. B_{prev} and $B_{current}$ are the previous and current opacity map buffers.

```
1.   $D_0$ = Min (Depth($p_j$)) for all $p_j$ in P ($1 \leq j \leq M$)
2.   for ($1 \leq i \leq N$)                                        (Loop 1)
3.       Determine the opacity map's depth $D_i$ from the light
4.   for each shadow sample point $p_j$ in P ($1 \leq j \leq M$)    (Loop 2)
5.       Find $i$ such that $D_{i-1} \leq Depth(p_j) < D_i$
6.       Add the point $p_j$ to $P_i$.
7.   Clear the alpha buffer and the opacity maps $B_{prev}$, $B_{current}$.
8.   for ($1 \leq i \leq N$)                                        (Loop 3)
9.       Swap $B_{prev}$ and $B_{current}$.
10.      Render the scene clipping it with $D_{i-1}$ and $D_i$.
11.      Read back the alpha buffer to $B_{current}$.
12.      for each shadow sample point $p_k$ in $P_i$                (Loop 4)
13.            $\Omega_{prev} = sample(B_{prev}, p_k)$
14.            $\Omega_{current} = sample(B_{current}, p_k)$
15.            $\Omega$ = interpolate (Depth($p_k$), $D_{i-1}$, $D_i$, $\Omega_{prev}$, $\Omega_{current}$)
16.            $\tau(p_k) = e^{-\kappa\Omega}$
```

Table 6-1. Pseudo code for opacity shadow maps algorithm

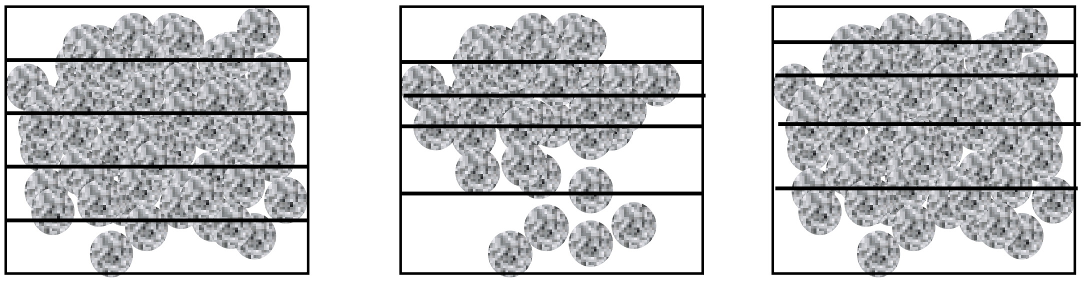


Figure 6-4. Choice of slicing schemes. (a) Uniform slicing (b) Density based slicing (1D BSP) (c) Non-uniform slicing similar to Gamma Correction.

In loop 1, the depth of each map is determined. Uniform slice spacing is reasonable for evenly distributed volumes (Figure 6-4a). When the structure of the volume is known, adaptive slicing (1D BSP) can be used such that tighter spacing is used in denser or more detailed regions (Figure 6-4b). Considering that regions farther from the light have ever-decreasing variations of shadows, a nonlinear partition conforms to perceptual sensitivity analogous to gamma correction (Figure 6-4c).

Prior to shadow rendering, shadow samples are produced. In volumetric clusters, the primitives tend to be very small and thus the end points of lines and the vertices of polygons often suffice. Thus, for each hair strand, we choose the endpoints of line segments used for local shading as shadow samples. More samples can be taken if needed. When many samples are required for each primitive, it may be useful to pre-compute the visibility and use only the visible points as shadow samples. Loop 2 prepares a list of shadow samples that belong to each buffer. The procedure makes the shadow computation time linear in the number of samples.

Each pixel in the map stores the opacity value, which is a summation that produces the integral term Ω in equation (6-2). Thus each primitive can be rendered antialiased with hardware support in any order[12]. The alpha buffer is accumulated each time the volume is drawn with the OpenGL blend mode *glBlendFunc(GL_ONE,GL_ONE).* The depth buffer is disabled. Clipping in

[12] The order can be arbitrary due to the commutative nature of addition (or integration).

line 10 ensures correct contribution of alpha values from the primitives and culls most primitives, speeding up the algorithm.

As loop 3 and 4 use only two opacity map buffers at a time, the memory requirement is independent of the total number of opacity maps computed. In loop 4, the shadow is computed only once for each sample. So, the amortized cost of the algorithm is linear in the number of samples. The overall complexity is *O(NM)* since the scene is rendered for each map, but the rendering cost is low with hardware acceleration. With scene culling scheme, the observed complexity is close to *O(N)*

The sample function in loop 4 can be any standard pixel sampling function such as a box filter, or higher-order filters such as Bartlett filter and Gaussian filter [Foley 1995]. For the examples shown in this chapter, a 3x3 averaging kernel is used. Such filtering is possible because alpha values are stored instead of depths. The sampled opacity values $\Omega_{prev,}$ $\Omega_{current}$ are linearly interpolated for each point p_k.

$$\Omega(\mathrm{p_k}) = t\Omega_{\mathrm{current}} + (1.0 - t)\Omega_{\mathrm{prev}}, t = (Depth(p_k) - D_{i-1})/(D_{\mathrm{i}} - D_{\mathrm{i-1}}) \quad (6\text{-}2)$$

A higher order interpolation may be used. For example, four buffers will be needed for a cubic-spline interpolation.

A volume turns opaque as the opacity Ω reaches infinity. The quantization in the alpha channel limits the maximum amount of opacity that a pixel can represent. A constant κ in line 16 controls the scaling of opacity values such that $e^{-\kappa} = 2^{-d}$, where d is the number of bits per pixel (for example, κ is about 5.56 for 8 bit alpha buffer). Thus, an opacity value of 1.0 represents a complete opaqueness. The transmittance function for an opaque surface is a step function [Lokovic 2000]. Although step functions are approximated with a large number of maps, the integration of opaque objects is more efficiently achieved by adding a traditional depth buffer shadow map (Figure 6-5). Figure 6-5 shows selected opacity maps.

For shadowed shading, the color for each end point for each polyline is computed as

$$\Psi_{Hair} = \Psi_{Ambient} + \tau(\Psi_{Diffuse} + \Psi_{Specular}) \tag{6-3}$$

, where τ is the computed transmittance approximation from (6-2), and the shading terms are computed with the shading functions described in section 5.4.

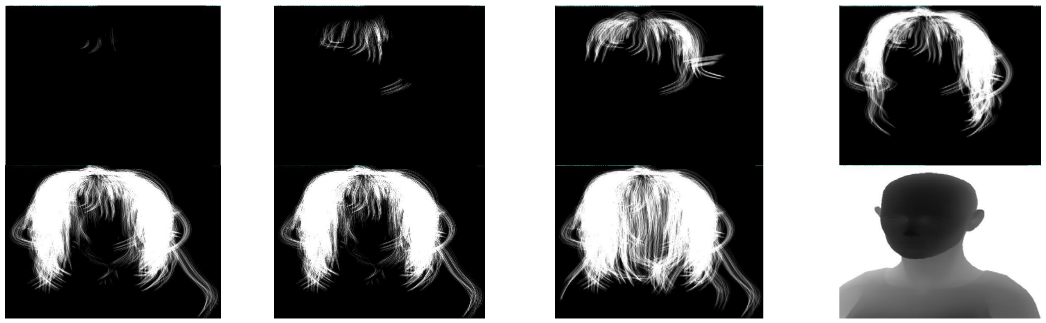


Figure 6-5. Opacity shadow maps. Each *i*th map is shown from left to right, top to bottom (*i* = 1,6,14,26,40,52,60). The last figure shows a depth map for the opaque head and torso.

6.2.2 Result and Performance

The performance of the algorithm depends on three factors – the complexity of model (M), the number of maps (N) and the resolution of each map. The performance is tested with a hair model of about 340,000 lines (Figure 6-6). Opacity maps are created at a resolution of 512 x 512 with uniform slicing scheme. The rendering time is linear in the number of the maps (Figure 6-7).

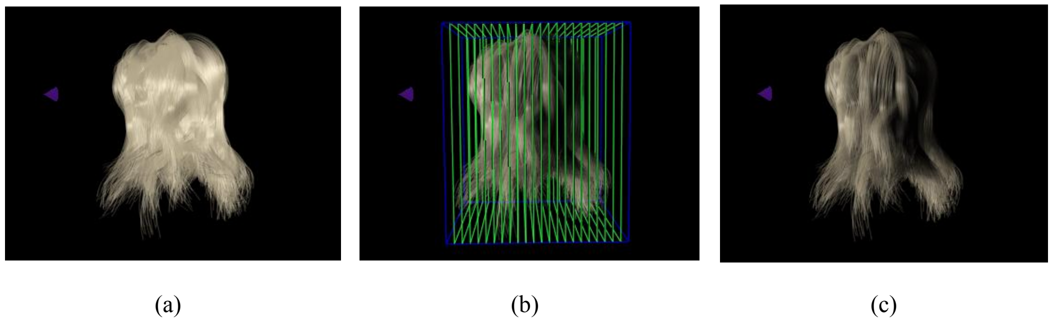
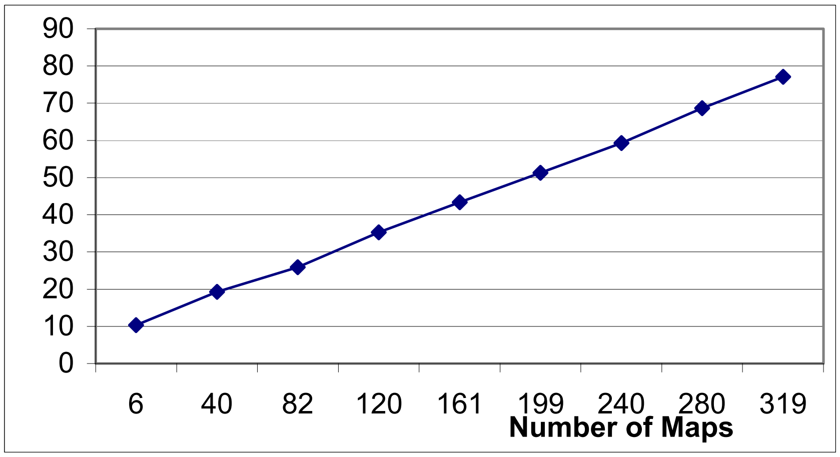


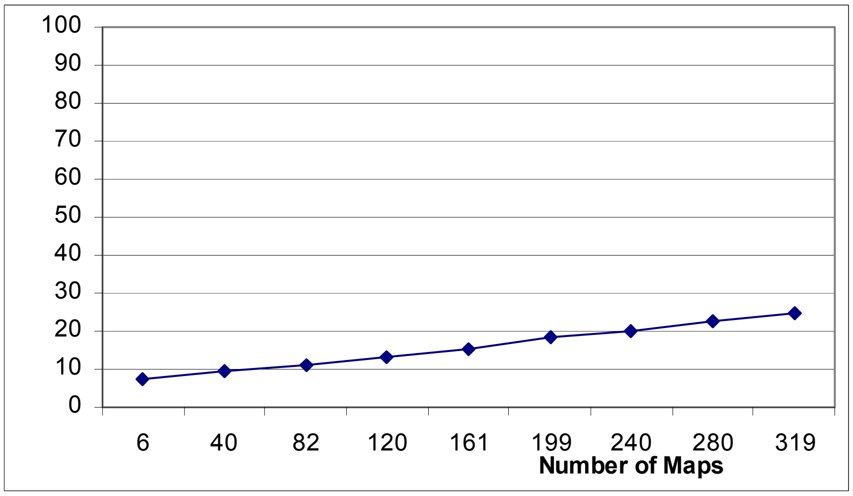
Figure 6-6: (a): Hair rendering without self-shadows. One directional light is positioned in front of the model (left side of the image) (b) Each opacity map is illustrated as a green rectangle. (c) Shadowed hair model (about 340,000 lines).

Loop 3 and 4 account for most of the calculation time. Other steps such as the bounding box calculation (0.09 sec), assigning shadow samples to the maps (2.21 sec), and object shadow map generation (0.24 sec) use constant time.

The slope of the graph in Figure 6-7 indicates the hardware's performance. About 4 opacity maps per second are computed using an SGI 540 NT Workstation with a 550Mhz Pentium CPU that can render about five million triangles per second. The performance increases in accordance with the transfer rate of the graphics card, rendering rate, and the speed of the CPU (Figure 6-7b).

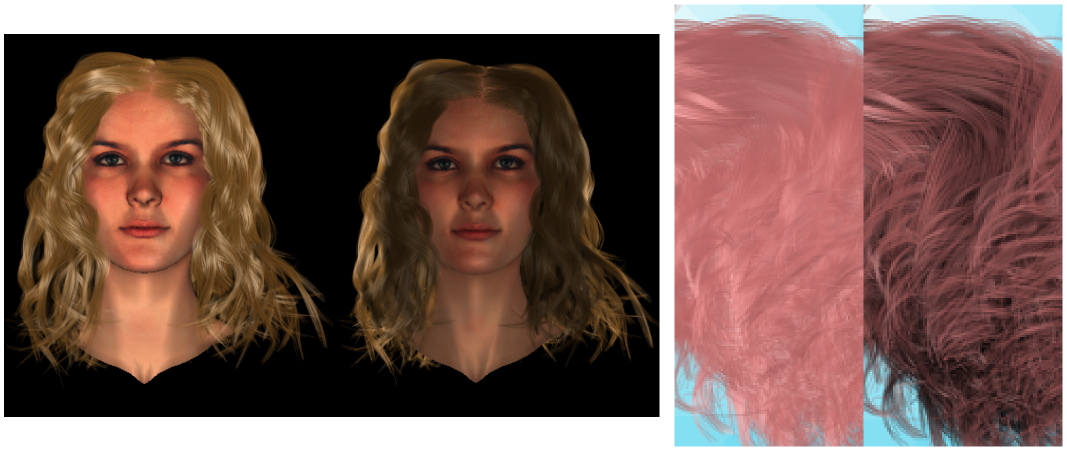


(a)



(b)

Figure 6-7 Rendering time measured in seconds. (a) Pentium 550 Mhz with PCI cobalt card (5M triangles /sec) (b) Pentium 700 Mhz with AGP nVidia GeForce3 (30M triangles /sec)



(a) (b)

Figure 6-8. Effects of shadows. (a) left: front lighting right: back lighting. (b) left: without shadows, right: with shadows

When the map resolution is small, the rendering rats (O(M) term) dominates the overall performance. On the other hand, when the map resolution is large, the buffer reading cost dominates the performance (O(N) term).

Figure 6-8a shows hair models rendered at various lighting conditions. Figure 6-8b shows a complex hair model rendered with and without shadows. In Figure 6-9, opacity shadow maps were used for rendering animal fur.

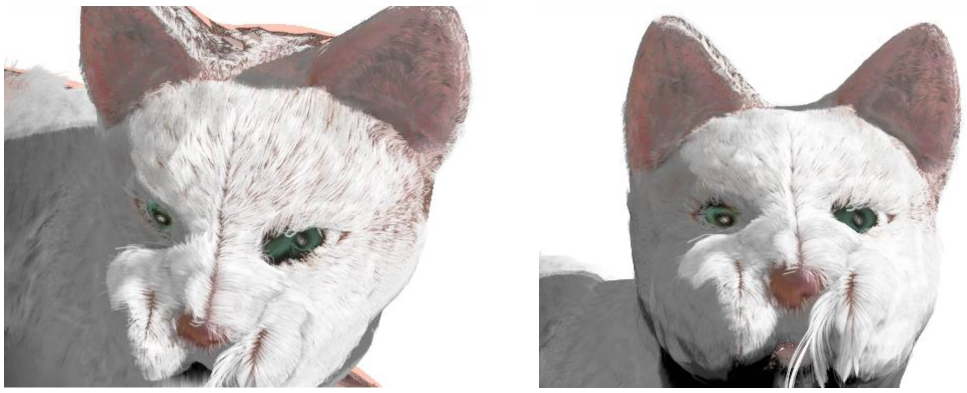


Figure 6-9. Rendering of animal fur

6.2.3 Discussion

Opacity shadow maps (OSM) provide a fast and memory efficient solution to rendering time-varying volumetric self-shadows such as hair animation without requiring any expensive preprocessing. OSM also provides the tradeoff between the speed and the quality by varying the number of maps. Thus, the user can quickly attain a preview of the shadow rendering, helping the light setup. For example, in an interactive hair modeling framework (chapter 4), using small number of opacity maps with low complexity hair model (section 4.7) enables interactive shadow updates during modeling. On the other hand, increasing the map resolution and the number of maps improves the quality of shadows.

For more efficient shadow updates, the hair volume could be spatially partitioned. Instead of drawing every volume, the line 7 of the algorithm can be rewritten as 'load the stored alpha buffer for the volumes closer to the light'. The partitioning scheme can improve the performance and reduce memory requirements for the scene graph if OSM is used in an offline renderer.

In a complex geometry such as hairs, the self-shadowing procedure can be viewed as a convolution of two high frequency signals, one from the geometry, and the other from the shadows. The study of the human visual system indicates that humans do not easily separate one signal from such mixed signals [Bolin 1995]. OSM is essentially a low-pass filter for shadow signals. Due to

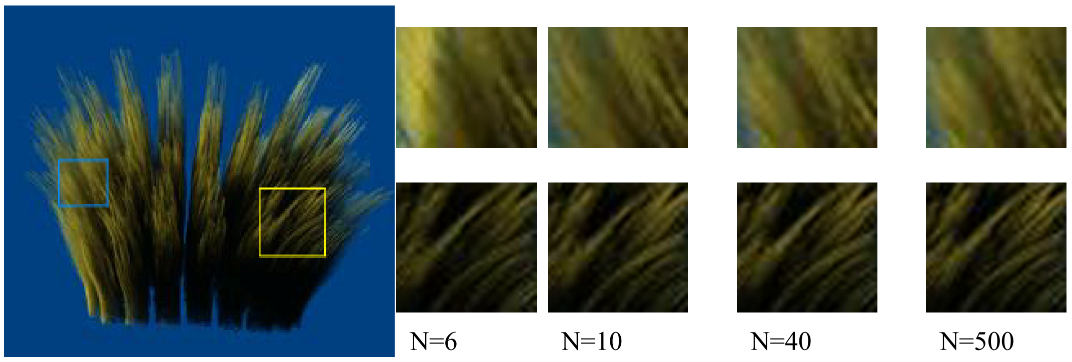


Figure 6-10: A fur model rendered with 500 maps (left). One light is positioned at the left side of the image. Close-up views are shown on the right side, upper rows for a bright area (blue rectangle) and lower rows for a dark area (yellow rectangle). Note that artifacts are visible at brighter regions, but only with relatively small number of maps (N = 6).

the perceptual effects of visual masking, the degradation in perceived quality may be lower than quantitative measures might predict (Figure 6-10).

There can be two sources of temporal aliasing in OSM. The piecewise linear approximation of opacity values can cause discontinuity in map boundaries, resulting in temporal aliasing with a small number of maps. Also, some low cost graphics cards do not support line antialiasing in hardware. Poor sampling of thin line can cause aliasing in generation of opacity maps. In our experiment, the higher order interpolation scheme combined with hardware antialiasing best approximates low-pass filter for shadow.

6.2.4 Comparison with Other Methods

Deep shadow maps (DSM) [Lokovic 2000] and OSM approximate the same property (transmittance) in different ways. DSM was originally developed in the context of scan conversion based software rendering system such as the photo-realistic RenderMan system [Upstill 1992]. The special data structure (linked list) is difficult to construct in any hardware-based approach. A good property of DSM is that once the shadow maps are created, the scene need not be stored at rendering steps. On the other hand, opacity maps require the scene graph to reside in memory all the time. This can be considered a weakness of OSM. However, OSM is targeted for time-varying volumes such as in hair animation. In the case volume does not change in time, OSM can be used as a preprocess for deep shadow maps, since the two methods essentially approximate the same transmittance.

Another commonly used method for rendering volumetric shadow is volume ray tracing (ray marching [Jensen 2001a]). Although volume ray tracing can be effective for relatively smooth and continuous medium such as clouds or smokes, the full volume ray tracing of hairs can incur high memory requirement to store voxel data. For example, in [Kong 1999], visible volume buffers are used to approximate shadows. Even with a resolution of 512^3, the resulting shadows are subject to aliasing. Also the rendering time and data initialization time rapidly increases as the resolution of volume increases. Table 6-2 and Table 6-3 compare opacity shadow maps with other methods.

	Deep shadow maps	Opacity shadow maps
Generation method	Software (CPU)	Hardware
Number of passes	Single pass	Multiple passes (N-pass)
Computation time	O(MlogM)	O(M)
Memory	O($M^{5/4}$)	O(M) (+scene graph)

Table 6-2. Comparison with deep shadow maps. M: number of samples

	Volume Rendering	Opacity Shadow Maps
Data Structure	Voxel data	Explicit geometry
Memory requirement	O(M^3)	O(M^2)
Precision	Voxel space accuracy	Image space accuracy
Preprocessing	Data conversion to voxels every frame.	No preprocessing

Table 6-3. Comparison with volume rendering

6.2.5 Extensions

As discussed in chapter 5, a hair fiber can scatter incoming lights in all the directions. Although the opacity shadow maps algorithm effectively handles forward scattering effect, multiple scattering is not accounted for in this method. Also, the shadow maps consider only single source light such as point sources, or directional lights. Soft shadows due to large area source light and natural illumination are not simulated with opacity shadow maps (or deep shadow maps). Figure 6-11 illustrates types of scattering to be handled in hair shadows.

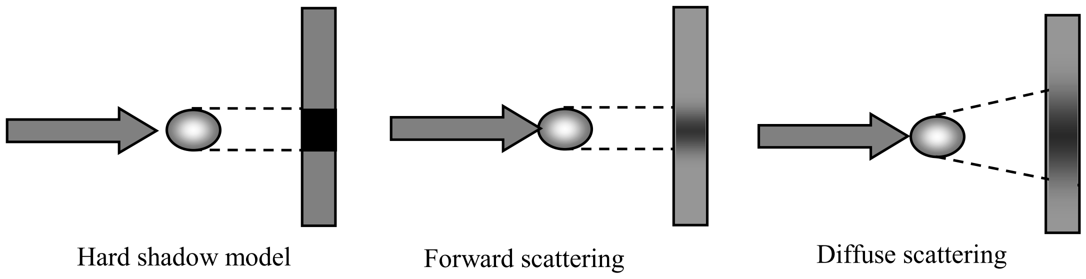


Figure 6-11. Different types of scattering that affect shadow pattern.

6.3 Anitialiasing

Hair strands are so thin (0.03mm to 0.09mm) that they can produce a high frequency image in a normal viewing condition. The projected thickness of a hair strand in image space is often much smaller than the size of a pixel. This causes serious sampling problems called *aliasing*. Without correct sampling and filtering, a line drawing looks jagged (Figure 6-12a). The fraction of the pixel covered by the line should be correctly computed and weighted (Figure 6-12b). Many current graphics hardware can perform antialiased line drawing. However, a depth buffer contention problem happens when multiple lines overlap each other. One line fragment takes the ownership of the entire pixel, preventing other fragments from being drawn. This results in haloed lines, instead of correct blending of colors. Figure 6-11c illustrates results of hair rendering with and without antialiasing.

Recall from section 5.7 that, for correct blending, the contributions of all the hairs should be integrated as (equation (5-30)),

$$I_{i,j} = \int_{-r}^{r}\int_{-r}^{r} f(s,t)\psi(i+1/2-s, j+1/2-t)\,ds\,dt$$

Supersampling such as the accumulation buffer [McReynolds 1997] is an easy solution to this integration problem, but at the cost of increased time. As noted in section 5.1, the number of samples for hair rendering should be sufficiently large and the rendering time increases linearly

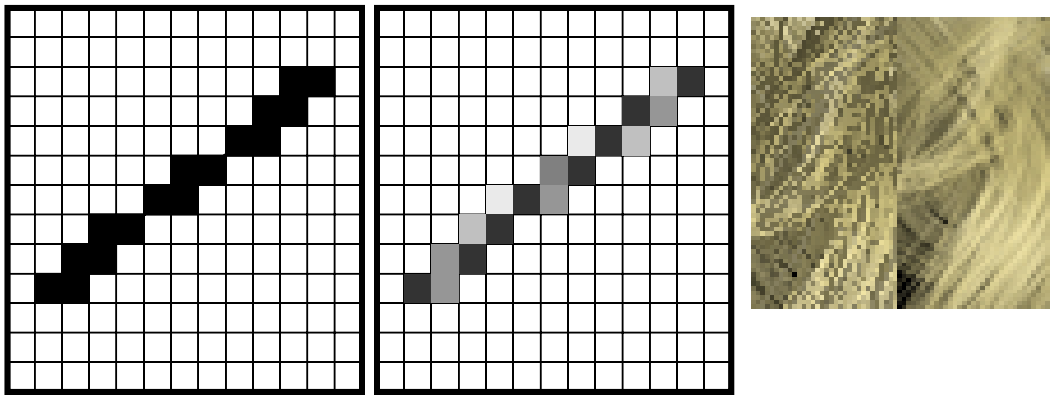


Figure 6-12. (a) Aliased line drawing (b) Correctly filtered line drawing (c) Left: hair rendering without any anitialiasing Right: hair rendering with antialiasing.

with the number of samples. Moreover, the antialiased line drawing option in OpenGL cannot be used since the correct result depends on the drawing order [McReynolds 1997].

Another way to compute the integration is the alpha blending [LeBlanc 1991]. Alpha blending requires correct visibility order (Figure 6-13). For each pixel, a list of primitives that cover any fraction of the pixel is created. These primitives are sorted in depth and the color of each primitive is blended from back to front order. The visibility order should be computed for every pixel in the image, an operation that can be expensive. Since the alpha blending should be done in software, the hardware accelerated line antialiasing is not utilized.

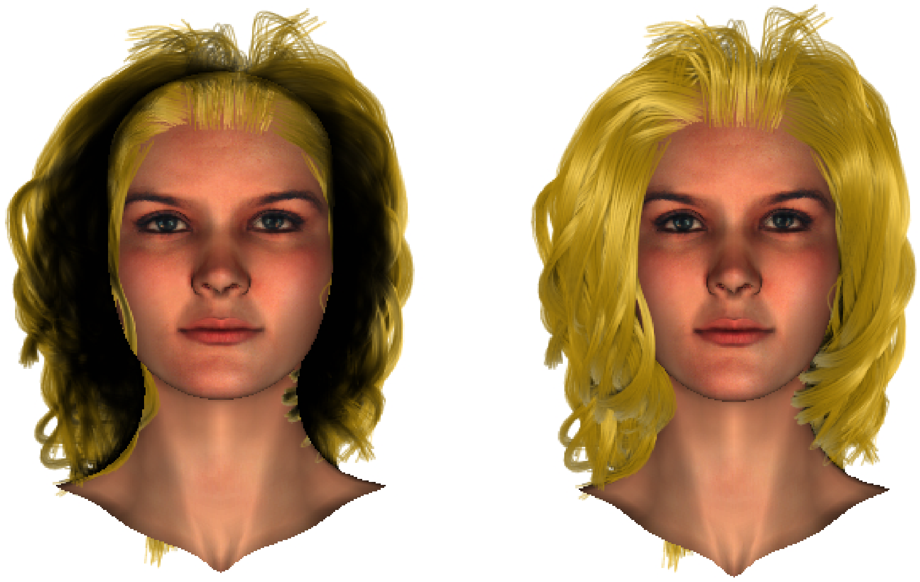


Figure 6-13. Alpha blending needs a correct drawing order. Left: incorrect visibility order. Right: correct one.

Alternatively, each line segment can be thought of as a thin rectangular polygon and existing visibility ordering algorithms for polygons may be employed [Newell 1972] [Snyder 1998]. However, these algorithms are targeted for polygonal models composed of hundreds to thousands polygons. Considering the number of hair strands (10^4~10^5), and sampling along each hair strand (10^2), the complexity may not be practical for these methods.

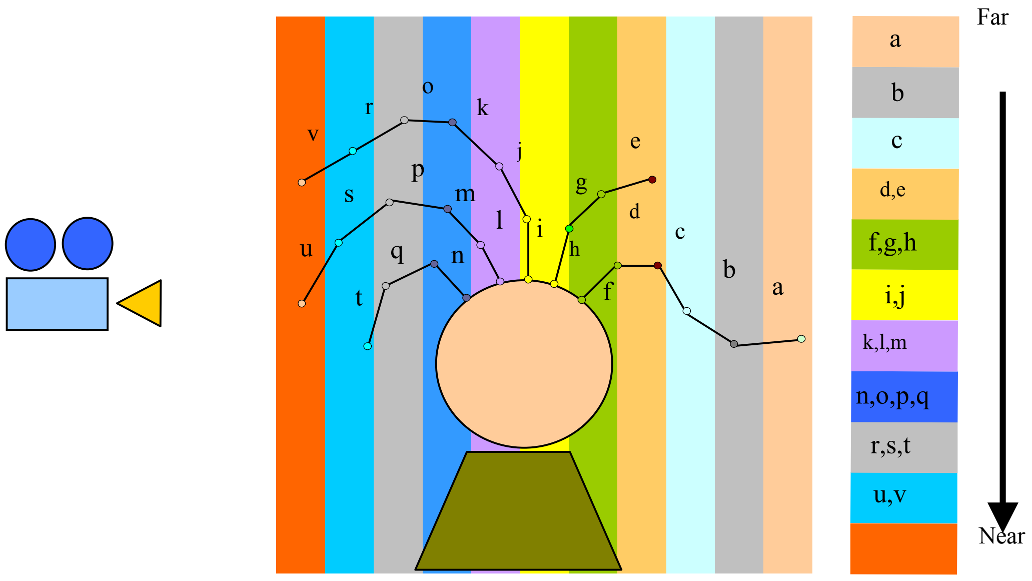


Figure 6-14. The polyline segments are ordered by the distance from the camera.

6.3.1 A Simple Visibility Ordering Algorithm

Antialiasing is performed in two steps. First, the visibility order of a given hair model is determined based on the distance to the camera (Figure 6-14). The bounding box of all the segments is sliced with planes perpendicular to the camera. Each *bin*, a volume bounded by a pair of adjacent planes (drawn as a colored bar in Figure 6-14), stores indices of segments whose farthest end point is contained by the bin. After other objects (e.g., a head mesh) are drawn, the depth buffer update is disabled. Then, the segments are drawn as antialiased lines such that the ones indexed by the farthest bin are drawn first.

The end points of polyline segments are grouped into a set of bins that slice the entire volume (Figure 6-14). The slab enclosing each point is found by

$$i = \left\lfloor N \frac{D - D_{\min}}{D_{\max} - D_{\min} + \varepsilon} \right\rfloor, 0 \le i < N \tag{6-3}$$

, where *i* is the index for the bin and *N* is the number of bin. D is the distance from the point to the image plane. D_{min} and D_{max} are minimum and maximum of such distances, respectively. ε is a constant such that $\varepsilon < \frac{D_{\max} - D_{\min}}{N}$.

Given a point $\vec{p}$ and the camera positioned at $\vec{c}$ looking in direction of $\vec{d}$, the distance to the image plane is computed by

$$D = (\vec{p} - \vec{c}) \cdot \vec{d} \tag{6-4}$$

Proper visibility ordering of lines is difficult to obtain by depth sorting alone. When lines are nearly parallel to the image plane, the errors are small, provided the spacing between bins is dense enough. However, when line segments extend over many bins, the visibility order cannot be determined either by maximum depth or minimum depth. Such lines could be further subdivided. However, on the other hand, the pixel coverage of such a line tends to be small. For example, when a line is perpendicular to the image plane, the pixel coverage of the line is just a single pixel. From our experiment, using maximum depth for every line produces good results.

In drawing pass, each bin is drawn from the farthest to the closest. For each bin, the line segments whose farther points belong to the slab are drawn. Each line segment is drawn with hardware antialiasing. The color of each line segment is accumulated using

```
glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA)
```

Although simple, the method is fast and converges to exact ordering as hair strands are drawn with more segments. The visibility-ordering algorithm runs at about 700,000 lines per second on a Pentium III 700Mhz CPU. Note that the algorithm shares many aspects with the shadow algorithm in the previous section, and it also makes full use of graphics hardware, if available. Since we keep relatively dense line segments for each hair strand, the algorithm produces visually satisfactory results. In the interactive modeling framework, the viewpoint does not change much from frame to frame (Figure 6-15). This coherence enables performing the visibility ordering periodically, not at every frame. In contrast, depth buffer based super-sampling methods (e.g., accumulation buffer [McReynolds 1997]) must compute visibility at every frame.

In addition, the algorithm is independent of screen resolution, in contrast to the supersampling method or other alpha blending method that computes the visibility order on pixel-basis. As with any alpha blending method, the alpha values of segments can control the perceived thickness of hair strands. As strands become thinner, super-sampling methods would require more samples while alpha value changes suffice in the visibility-ordered hair model (Figure 6-16).

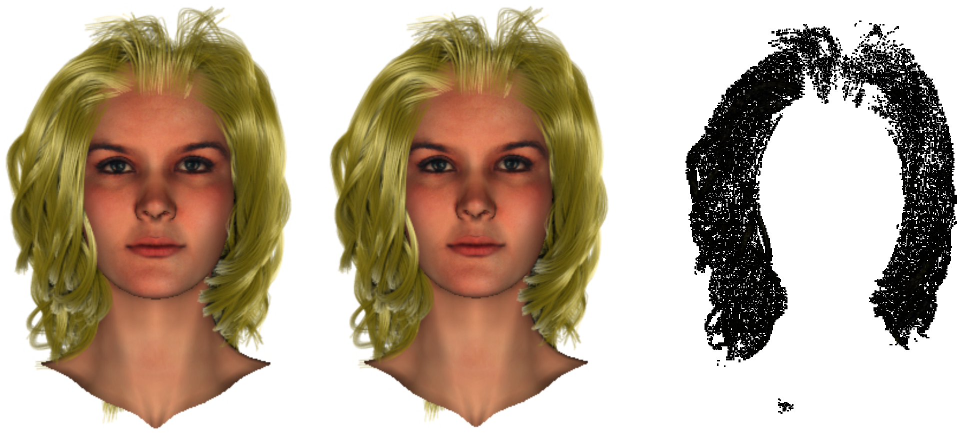


Figure 6-15. Coherence of the visibility order. The image on the left was rendered using the visibility order from a viewpoint that was rotated by about 30 degree from the original viewpoint. The image in the middle was rendered with the correct visibility order. Pixel value differences are shown on the right. Although there are differences in the pixel values, these differences do not usually produce noticeable visual artifacts.

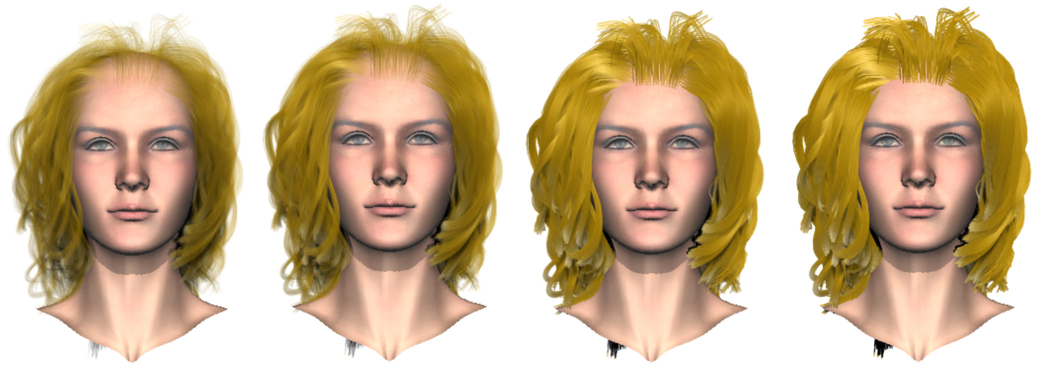


Figure 6-16. Effects of different alpha values. From left to right: $\alpha = 0.09, 0.25, 0.60, 1.00$.

6.3.2 Discussion

Note that by increasing the number of line segments for each hair strand, one could achieve the exact visibility ordering. For example, if the length of each line segment is roughly equal to the width of each pixel, the visibility ordering is equivalent to sorting hair fragments per each pixel, as was done by [LeBlanc 1991], [Chang 2002]. Thus, as in the case of shadow rendering, the tradeoff between the quality and the speed of rendering can be controlled with the number of line segments per hair.

A benefit of the visibility ordering algorithm presented in this section is that one can fully utilize existing graphics hardware. Once the visibility order is computed, antialiased hair rendering is no less efficient than standard rendering method. One current limitation is that the pixel sampling is essentially through a box filter. A better reconstruction kernel could improve the image quality [Smith 1995]. Recent advances in graphics hardware promise that better reconstruction filters could be also incorporated in hardware for antialiased hair rendering in the near future [Deering 2002].

6.4 Conclusion

In this chapter, a set of hair rendering algorithms was described in detail. The polyline-based approach enables efficient hair rendering through the heavy use of existing graphics hardware. Currently, the limiting factor for the image quality is the memory constraint imposed by explicit hair models. In reality, each hair strand is essentially a smooth and curved object. However, in practice, the discrete approximation of each hair strand as a polyline produces a large number of line segments. The number of line segments (and hence the length of each line segment) is closely related to the accuracy of the shading calculation, shadow quality, and antialiaing, as well as the accuracy of the model shape. Increasing the number of line segments improves the image quality, but also increases the memory requirements to store all the intermediate end points.

The following recent advanced features in graphics hardware could be further exploited to increase the performance and the image quality and also to reduce the memory requirement. Direct use of spline curve rendering supported in OpenGL will eliminate the need of intermediate polyline points that incur high memory requirement. However, for this option to be used in hair rendering, the followings also should be considered concurrently.

- Shading calculation should be done inside the graphics pipeline – this may require the use of the programmable vertex/pixel shader [Lindholm 2001] [Upstill 1992]
- Shadows should be also accessible from the graphics pipeline – this may require the use of 3D textures to store shadow values. Having the 'render-to-texture' option [Möller 1999] would eliminate the high memory bandwidth between the GPU and main CPU to read the opacity maps back to main memory.
- Antialiasing should be also supported in hardware – either through supersampling [Deering 2002], or some new methods.

Although the current set of hardware does not meet all these requirements, the next generation graphics hardware might support all these functionality in the near future. This may lead to high quality real-time rendering of arbitrary hair models. There also exist avenues for extending the current rendering algorithms for better image quality. Throughout this chapter, many simplifications and approximations were made, to improve the speed of rendering algorithms to be used in an interactive rendering system. For better image quality, the followings may be considered.

- Any shadow map approach essentially assumes a controlled lighting environment. More general approach to compute shadows is needed to render more complex lighting situation, such as natural illumination.

- Due to its implied computational expense, multiple scattering effects are ignored in existing hair rendering research. However, for some types of highly scattering hair fibers (e.g. blond), multiple scattering may prove essential for photorealistic hair image synthesis.
- Combining above two, an efficient global illumination algorithm is needed. Recently, global illumination proved essential in rendering natural objects such as skin [Jensen 2001b]. Further investigation might yield an efficient hybrid algorithm that exploits advanced graphics hardware for realistic hair rendering.

This chapter concludes the report of my research work on hair rendering, especially aiming at developing an interactive hair rendering system that exploits existing graphics hardware as much as possible, for both the increased performance and improved image quality in hair rendering.

Chapter 7

7. Animation and Future work

This thesis focused mainly on two topics – modeling the structural aspect of human hair, and rendering the hair model. In this chapter, another important aspect of hair, animation, is discussed in detail. Although the main contributions of this thesis are not on animation, the challenges and findings in my own attempts for developing methods for hair animation are discussed in section 7.1 through 7.3. Ideas on extending the current modeling system are conceptualized in section 7.4. Finally, possible avenues for various future extensions are desc ribed in section 7.5.

7.1 Towards Interactive Hair Animation

In an attempt to incorporate a dynamics engine into the interactive modeling system, I built a simple system illustrated in Figure 7-1.

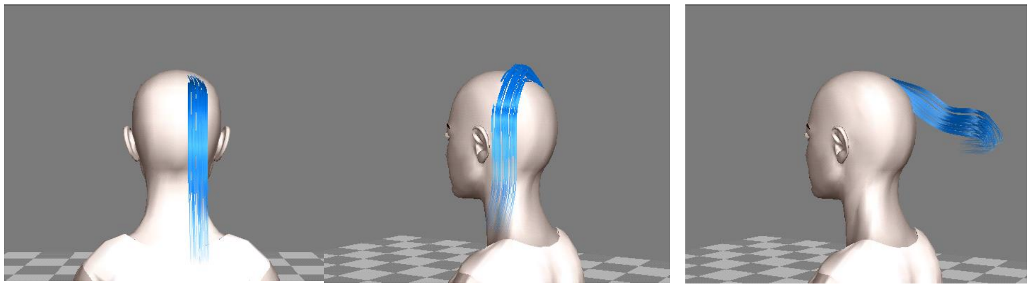


Figure 7-1. Interactive hair modeling assisted by hair dynamics.

In this example, the user grabs a hair wisp similar to the layered model [Plante 2001] and interactively manipulates the shape and position of the wisp by pulling it around. There are several issues before this simple system can be really useful for full hair animation or interactive modeling. Accurately simulating the motion of hair involves many issues described below.

- An efficient method is required to simulate the motion of each hair strand (or the skeleton of a wisp). Hair strands not only move freely but also deform on collisions between hair and other objects such as scalp and shoulder.
- Collisions should be detected and correctly handled. Hairs should not penetrate other parts of human body as well as other hairs.
- Hair is a viscous, deformable volume. The static charges and frictional forces often cause hairs to form a cluster. These large clusters often behave similarly to rigid bodies, but upon excessive forces or collisions, hairs tend to break and split away. Methods to simulate these complex interactions are needed.
- An animation technique should not be limited to a specific class of hairstyles.
- Often, hairstyles are restored to their original shape when external forces are removed. However, upon excessive forces, hair tends to lose its shape memory and deform into new shapes.

At the time of this thesis writing, there do not seem to exist any animation technique that can handle all these issues. Nevertheless, there are some notable advances in hair simulation. Each technique focuses on different aspects of the problem. In the following sections, I will discuss each aspect of hair animation.

7.2 Hair Strand Dynamics

Early work on hair animation mostly ignores the interactions between hairs and considers each hair strand as an independently moving entity [Anjyo 1992] [Daldegan 1993] [Kurihara 1993] [Rosenblum 1991]. The main focus is to simulate the motion of each individual hair strand. A hair strand is a very pliable and deformable object. There can be three types of deformations – bending, twisting, and stretching. Among these effects, bending is the main source of hair motion. Each hair strand strongly resists stretching or shearing, but gets easily bent. Also, each hair strand tends to restore its shape when the cause of bending is removed.

7.2.1 Cantilever Beam and One Dimensional Angular Dynamics

The cantilever beam approach, introduced by Anjyo et al., treats each hair strand as a set of linked rigid rods [Anjyo 1992]. The static pose of each hair strand at rest is found by cantilever beam simulation for hair bending. Based on a simple formula describing the bending moment of each section of hair strand, the method computes the amount of bending for each section relative to its previous section. Hair motion is simulated using one-dimensional projective equations for hair dynamics. Simple ordinary differential equations are defined for two polar angles for each section. The external force is separately applied to each projected component of motion (azimuth and zenith angles). The motion is processed from top to bottom, where the position of the root is first processed and the positions of lower parts of the hair strand are recursively defined. This method is later used by Lee and Ko [Lee 2001], where they add collision handling schemes.

7.2.2 Forward Kinematics

More general alternatives for one-dimensional projective equations are the forward kinematics techniques developed in the field of robotics [Chang 2002] [Hadap 2001] [Pai 2002]. Hadap and Thalmann, for example, formulate the motion of each hair strand using the generalized coordinate formulation [Hadap 2001]. A hair strand is represented as a serial rigid multi-body chain. The motion of each hair strand is simulated with forward kinematic equations through the reduced coordinate formulation. A similar method is used in [Chang 2002], where the generalized coordinate formulation is used for each key hair strand that approximates the continuum model.

Since a hair strand is very thin, it has very large tensile strength compared to its bending and torsional regidity. The strong tensile mode of motion can cause stiff set of equations that can be numerically unstable. The benefit of using the forward kinematics method is that this stiffness can be avoided through the reduced coordinate formulation. On the other hand, interaction with environments should be formulated in terms of external forces. In some cases, correct handling of collision involves solving for the inverse kinematic equations, which can be costly to compute. For example, if the tip of long hair hits the shoulder, the entire hair strand should be accordingly bent due to the collision force. This type of response is difficult to formulate with forward kinematics techniques, as pointed out by [Plante 2001].

A more involved treatment for the simulation of thin solids is introduced in [Pai 2002]. The method is based on the Cosserat theory of elastic rods. The physical model reduces to a system of spatial ordinary differential equations that can be solved efficiently for typical boundary conditions. The model handles the important geometric non-linearity due to large changes in shape. It is demonstrated that the model can be used for interactive simulation of surgical sutures. However, it is not yet clear if this model is efficient enough to be used in the context of hair simulation where a large number of hair strands should be simulated.

7.2.3 Mass-Spring Simulation

The mass-spring system was first used in [Rosenblum 1991] for hair animation. In this formulation, each hair strand is modeled as a chain of mass-spring system (Figure 7-2). Hair is modeled as a set of particles connected by tensile, bending, and torsional springs. Treating each particle as a point mass, one can setup a set of governing differential equations of motion and try integrating them. However, the original

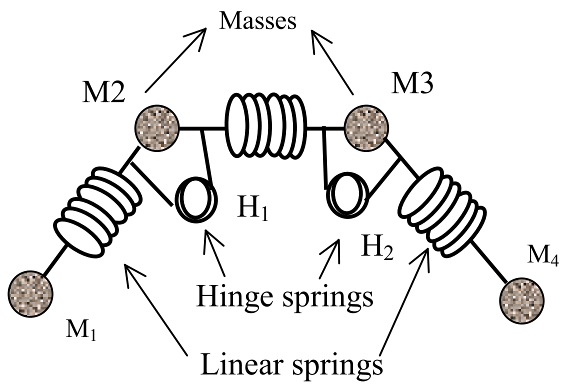


Figure 7-2. Mass-spring-hinge model

formulation by [Rosenblum 1991] suffered the stiffness problem – a hair strand strongly resists any linear stretching. Strong spring forces are required to simulate such stiffness, which in turn requires the time step size to be very small.

Recently, there were advances in mass-spring formulation, in the context of cloth simulation [Baraff 1998] [Desbrun 1999]. Implicit integration of the system continuously proves to be useful in a system involving stiffness. The implicit integration methods effectively smooth stiff linear spring forces, preventing the system from becoming unstable. A hair strand fiber could be simulated as one-dimensional set of mass-spring system [Fong 2001]. Implicit integration usually requires solving a linear system. For each hair strand, a linear system should be solved at each time step, which can be expensive. However, the specific system for hair has a block diagonal form that can be solved in a linear time. One benefit of mass spring formulation is that handling collision becomes straightforward. Another benefit is that one has more freedom in the connection topology than other methods. It is shown that a layered model simulating a hair wisp can be also formulated as a mass-spring system [Plante 2001].

In fact, there is no solution that outperforms other methods in every aspect. In general, articulated body and other methods based on angular dynamics have problems with handling collision. Collision handling is easier in mass-spring based simulation. On the other hand, angles are hard to preserve or constrain in these mass-spring systems. In contrast, articulated body simulation can easily achieve the torsional rigidity of a hair strand, so that a hair strand can retain its original shape.

7.3 Collision Detection and Interaction Handling

It can be visually distracting if a hair strand penetrates into other objects. Thus, a good collision detection algorithm is an essential starting point for a hair dynamics system. There exist various types of collision detection algorithms. To detect the collision between hairs and other scene objects, researchers have used use a simple ellipsoid-based pseudo force field [Anjyo 1992], head hull layer [Lee 2001], 3D grid structure [Plante 2001], and a fluid dynamics model based on smooth particles [Hadap 2001].

Note that there are two interesting phenomena associated with hair/hair interaction – clustering and fluidness (Figure 7-3). At large scales, the attraction forces due to static charge tend to make hairs stick together. These forces tend to make a hair cluster move as a group and collide with other clusters. At smaller scales, each hair strand move independently, behaving like a continuous fluid. There are always transitions between these scales. A large, coherently moving cluster will split into smaller ones upon excessive external forces, or collision. Conversely, a group of freely moving hair strands will merge into a larger cluster due to the attraction forces and static charges. The amount of these transitions is related to the material properties of individual's hair such as curliness, static charges, use of artificial styling products, etc. Smoother hairstyles will behave more like a continuous fluid, while hairstyles of many clusters will behave much like independent discontinuous rigid bodies. Many real hairstyles often have both properties mixed and the distinction between these two properties tends to be ambiguous.

Due to these complexities, hair/hair interaction has been mostly ignored in early work. Recently, researchers have developed a number of different schemes to handle hair/hair interaction for various model representations. Notable accomplishments include the layered wisp model [Plante 2001], the fluid dynamics model [Hadap 2001], and the key hair-based approach [Chang 2002].

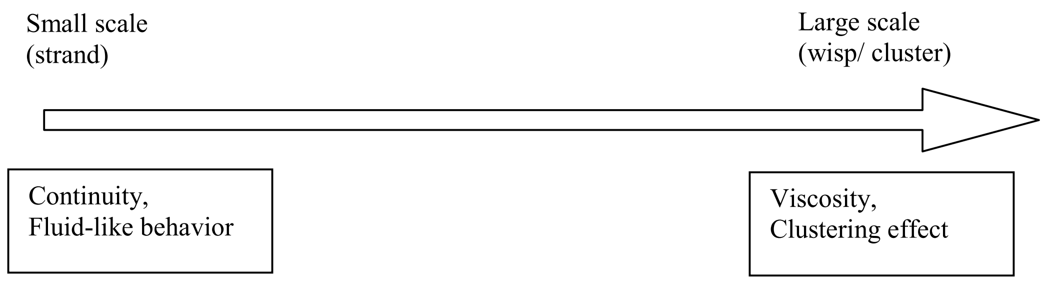


Figure 7-3. Two aspects of hair-hair interaction.

Hadap and Thalmann pioneered a radical approach to model dense dynamic hair as continuum by using a fluid dynamics model for hair motion [Hadap 2001]. Hair-hair collision is approximated by the pressure term in fluid dynamics while hair-hair friction is approximated by viscosity term. Single strand dynamics is solved using a multi-body open chain. Hair-air interaction is considered by integrating hair with an additional fluid dynamics system of air. This work presents a promising direction for the hair-hair interaction problem. Nonetheless, there are some limitations. The gradient may generate collision forces that are incompatible with the motion of hair since the pressure term has no knowledge of velocities of colliding hairs. Rather, these pressure terms are designed mainly to resist compression of the hair medium (e.g. stacking). This method can be very beneficial for smooth hairstyles that better obeys the continuum assumption. However, for more complex hairstyles, clustering effects are not accounted for and mechanisms for simulating discontinuous clusters are not considered. Without clustering effects, the animation may look too smooth.

The clustering effects are better captured with the layered wisp model [Plante 2001]. In this model, each hair cluster is represented as a deformable model and hair strands are interpolated inside each cluster. Each cluster is modeled as an anistropic, viscous volume. The interaction behaviors between clusters are formulated differently depending on their relative orientations. When clusters are parallel to each other, clusters are allowed to penetrate each other, while the clusters are not allowed to penetrate when they are of different orientations. The skeleton of the hair wisp captures the global deformation, while the envelope captures local radial deformations. This scheme is useful for the simulation of long, thick hair. However, the lack of coherence between nearby clusters may cause excessive collisions, which may be objectionable for smoother hairstyles. Hair clusters are pre-determined before animation. In reality, clustering occurs dynamically.

The mutual hair interaction model by Chang et al. is based on the key hair method, where a sparse set of guide (key) hairs is used to represent the entire hair volume [Chang 2002]. In their method, each key hair strand is modeled as a multibody open chain, similarly to [Hadap 2001]. A dynamic continuum model is developed on this sparse representation. Long-range connections among strands are modeled as breakable static links formulated as non-reversible positional springs. While small motions do not alter the hairstyle, excessive forces may break these springs permanently. Adaptive guide strands are generated on the fly and an auxiliary interpolation scheme is used to complement the sparse representation. The use of sparse key hairs significantly reduces the time needed to simulate hair-hair interaction, but also at the cost of reduced realism. Especially, even when the static links between key hairs are broken, the interpolation itself is not broken. These underlying interpolations scheme becomes increasingly problematic in case of complex hairstyles where discontinuity plays an important role for the appearance of hairstyle.

The real complexity of hair motion lies in the fact that there are continuous transitions between these types of hair-hair-interaction modes. A hair strand can move independently, moving away from other hairs. This *free* hair can suddenly join a big hair cluster and move together with other hairs. On the other hand, the coherently moving cluster can split into smaller clusters and many independently moving hair strands. Note that each technique mentioned above deals with the hair-hair interaction problem at each discrete level. The fluid continuum model [Hadap 2001] mainly views the problem at strand level, while the layered wisp model [Plante 2001] and key hair model [Chang 2002] mainly deals with the problem at cluster level. Realistic simulation of hair motion essentially involves integrating these aspects in a continuous manner in the spirit of the 'simulation level of detail systems' [O'Brien 2001]. One could formulate the hair-hair interaction problems in a multiresolution fashion, where dynamic clustering and splitting could be simulated. In this aspect, next section describes ideas on how one could extend the multiresolution hair model (chapter 4) for animation.

7.4 Multiresolution Framework for Hair Animation

This section proposes several options for extending the multiresolution hair representation for animation. At the highest level, the simplest way may be to provide the user with kinematics controls over root clusters. This way, the user can keyframe the motions of a small number of root clusters and all the other clusters at lower level will follow the motion the bind and update process. The motion of root clusters could be also generated through the layered wisp model [Hadap 2001]. Alternatively, existing strand based animation techniques [Anjyo 1991] [Hadap 2001] could be used to animate the lowest level strands. In this case, the restoration of the user defined hairstyles may become problematic since existing strand based techniques do not provide means of preserving the clustering effects.

Essentially, the key hair approach [Daldegan 1993] [Chang 2002] is just another form of representing hair clusters. It could be beneficial to combine key hair approaches with discontinuous wisp models. One could try fitting generalized cylinders between key hairs (e.g. four key hairs per each GC). This way, the continuous aspect of hair motion could be efficiently generated through key hair approach, while the multi-resolution hair model could maintain the necessary discontinuous clustering effects.

Ultimately, hair clusters should split and merge dynamically, automatically, and often partially. I anticipate that the dynamic clustering problem could be formulated as that of dynamically updating the hair tree, consistently changing the hair strand ownership. There seem two aspects in achieving this goal. In the current multiresolution modeling framework, the user specifies the level of subdivision for each hair cluster. In a top-down manner, this subdivision scheme could be automatically triggered based on conditions such as collision, the total mass of the cluster, extremity of the motion, etc. In a bottom-up manner, these subdivided clusters could be dynamically merged together based on conditions such as proximity of clusters, relative speed, compression, the user-defined degree of static attraction, etc. This way, existing hair-hair-interaction mechanisms could be used in a continuous manner. For example, when each hair strand moves independently, one could employ the fluid model [Hadap 2001], while upon some threshold, cluster-based method [Plante 2001] could be triggered and vice versa.

7.5 Future Work

There exist many avenues for future research in any aspect of human hair. Some samples of the issues were discussed in previous chapters. The remaining issues are listed below.

On the modeling side, it could be beneficial to develop a user controllable hair-hair interaction operator. For example, continuity becomes often difficult to enforce with cluster-based models [Yang 2000] [Plante 2001]. An efficient way to stitch the generalized cylinders to form a continuous volume may be desired. An alternative way may be to use a bottom-up approach, where generalized cylinders are fit to strand based models. This way, hair models generated from other modeling system could be easily imported and edited. Also, constructing a gallery of hairstyle database will greatly reduce the amount of time to design each hairstyle. Likewise, it could be desirable to transfer hair models from one face mesh to another. Developing a fill-in algorithm for sparsely defined hairstyles could be another worthy direction. For example, a sparse set of hair strands generated from photographs [Grabli 2002] could be used as the initial guideline for such algorithm. Matching structural details could be synthesized with the copy-and-paste operations described in chapter 4.

One the rendering side, existing interactively rendering algorithms could be further improved as discussed in chapter 6. Although photo-realistic rendering was the main goal of this thesis, it can be often desirable to reduce the visual realism based on the intent of visualization. For example, a few line drawings often suffice to illustrate a certain hairstyle (chapter 2). Figure 7-4 illustrates a simple test scene for NPR rendering of a hair model. Refer to [Gooch 2001] for more detailed discussions on NPR algorithms.

Figure 7-4. Non-photorealistic rendering of a ponytail hair model.

Since hair shares many aspects with other volumetric objects in nature (clouds, smokes, leaves, etc.), it could be worthwhile to apply hair rendering algorithms for such objects. There also exist other domains where similar properties are desired. For example, efficient hair rendering algorithms could benefit the scientific flow visualization problems where the trajectory of a flow filed is often drawn as smooth lines similar to hair strands.

This thesis mainly dealt with long human hair. Short hairs such as beard, or animal fur can be also handled in the same framework, but a simpler method may be preferred (see [Berney 2000] [Bruderlin 1999] [Fong 2001]). Figure 7-5 illustrates an example fur modeling procedure that was used to produce the animal fur models in chapter 6. A number of texture maps were used to specify properties such as hair density, hair shape, and curliness.

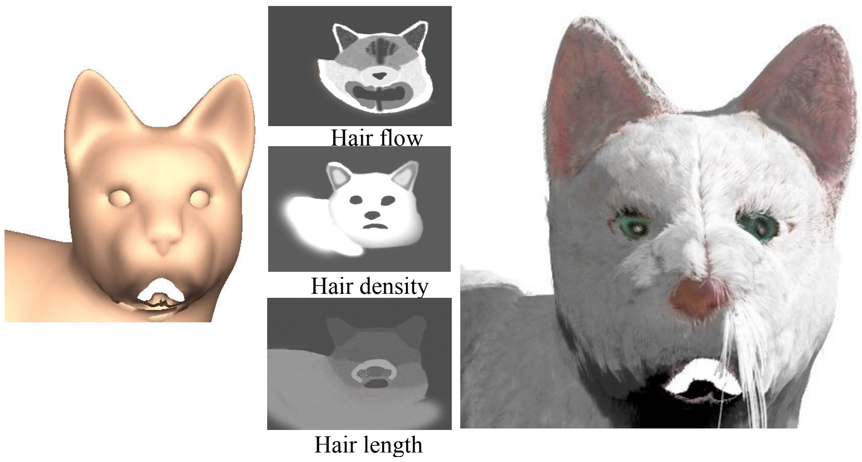


Figure 7-5. Modeling and rendering animal fur

7.6 Conclusion

Human hair presents challenges in every aspect of computer graphics. Each aspect (modeling, rendering, or animating) is a daunting task. At the beginning of my study, I was ambitious enough to address all these problems at the same time. Often, solving a single problem opened a dozen problems that I was not aware of. At times, I had to switch topics from modeling, to rendering, and to animation. Contributions made in this thesis grew out of a seemingly endless quest to find fundamental representations and algorithms for human hair, as illustrated in Figure 7-6.

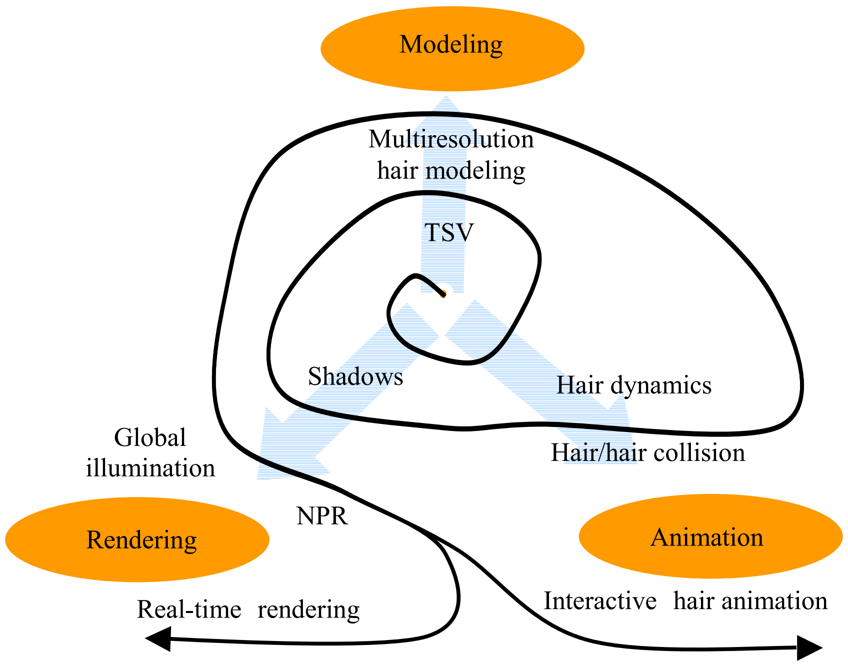


Figure 7-6. Progress spiral

I consider this thesis as just the starting point for numerous avenues for future research. Some of the issues were discussed in the thesis, but there are many issues not detailed here. To me, every problem in computer graphics is related to hair, and every single paper out there could potentially contribute to solving problems related to hair. I hope that this thesis could help motivate many other researchers to work on many aspects of human hair, building on the collective knowledge of this ever growing field of computer graphics.

Bibliography

[Aguado 1999] A.S. Aguado, E. Montiel and E. Zaluska. Modeling Generalized Cylinders via Fourier Morphing, ACM Transactions on Graphics, 1999, 18(4), 293-315

[Anjyo 1992] K. Anjyo, Y. Usami and T. Kurihahra. A simple method for extracting the natural beauty of hair, SIGGRAPH Proceedings, 1995, 111-120

[Banks 1994] D.C. Banks. Illumination in diverse codimensions, SIGGRAPH Proceedings, 1994, 327-334

[Baraff 1998] D. Baraff and A. Witkin. Large steps in cloth simulation. SIGGRAPH Proceedings, 1998, 43-54

[Berney 2000] J. Berney and J. Redd (organizers). Stuart Little: A Tale of Fur, Costumes, Performance, and Integration: Breathing Real Life Into a Digital Character, SIGGRAPH course note #14, 2000

[Blinn 1982] J. F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces, SIGGRAPH Proceedings, 1982, 21-29

[Bloomenthal 1990] J. Bloomenthal. Calculation of Reference Frames Along a Space Curve, Graphics Gems, Academic Press, 567-571

[Bolin 1995] M. Bolin and G. W. Meyer. A frequency based ray tracer, SIGGRAPH Proceedings, 1995, 409-418

[Bruderlin 1999] A. Brudelin. A Method to Generate Wet and Broken-up Animal Fur, Pacific Graphics, 1999

[Carlson 1986] B. Carlson, Communication Systems – an Introduction to Signals and Noise in Electrical Communication, McGraw-Hill, 1986

[Carpenter 1984] L. Carpenter. The A-Buffer, an Antialiased Hidden Surface Method, SIGGRAPH Proceedings, 1984, 103-108

[Chang 2002] J. Chang, J. Jin, and Y. Yu. A Practical Model for Mutual Hair Interactions, ACM Symposium on Computer Animation, 2002

[Chen 1999] L. Chen, S. Saeyor, H. Dohi, and M. Ishizuka. A system of 3D hair style synthesis based on the wisp model, The Visual Computer, 1999, vol. 15, 159-170

[Cormen 1989] T. Cormen, C. Leiserson, R. Rivest. Introduction to Algorithms, The MIT Press, 1989.

[Coons 1967] S. A. Coons. Surfaces for computer aided design of space forms, MIT Project TR-41, MIT, Cambridge, MA, 1967

[Crow 1977] F. Crow, The aliasing problem in computer generated shaded images, Comm. ACM, Vol. 20, No. 11, November 1977, pp 799-805

[Csuri 1979] C. Csuri, R. Hakathorn, R. Parent, W. Carlson and M. Howard. Towards an interactive high visual complexity animation system, SIGGRAPH Proceedings, 1979, 288-299

[Daldegan 1993] A. Daldegan, N. M. Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating, and rendering hair, Eurographics Proceedings, 1993, vol. 12, 211-221

[Deering 2002] M. Deering and D. Naegle. The SAGE Graphics Architecture. SIGGRAPH Proceedings, 2002, 683-692

[Desbrun 1999] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects, In Proceedings of Graphics Interface, 1999, 1-8

[Dippe 1985] M. Dippe, H. Erling, Antialiasing through stochastic sampling, SIGGRAPH Proceedings, 1985, 69-78

[Drebin 1988] R. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering, SIGGRAPH Proceedings, 1998, vol. 22, 65–74

[Duncan 2001] J. Duncan. Flesh for fantasy, Cinefex, July 2001, vol 86

[Durand 2000] F. Durand and J. Dorsey. Interactive Tone Mapping. Eurographics Workshop on Rendering, 2000, 219-230

[Falk 2001] R. Falk and L.R. Sande (organizers), “Shrek”: The Story Behind The Scene, SIGGRAPH Course Note 19, 2001

[Ferwerda 1988] Ferwerda, J.A. and Greenberg, D.P. A psychophysical approach to assessing the quality of antialiased images. IEEE Computer Graphics and Applications, 1988, 8(5), 85-95.

[Ferwerda 1998] Ferwerda, J. A. Fundamentals of spatial vision. Applications of visual perception in computer graphics, SIGGRAPH Course note #32, 1998, 1-27

[Folley1995] J. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. Computer graphics, principles and practice, Addison-Wesley, 1995, Second Edition

[Fong 2001] M. Fong. Animating monster fur, SIGGRAPH Course note 36, 2001

[Gelder 1997] A. V. Gelder and J. Wilhelms. An Interactive Fur Modeling Technique, Graphics Interface, 1997, 181-188

[Glassner 1989] A. S. Glassner. An introduction to ray tracing, Academic Press, 1989

[Glassner 1995] A. S. Glassner. Principles of digital image synthesis, Morgan Kaufmann, 1995

[Goldmann 1997] D. Goldman. Fake fur rendering, SIGGRAPH Proceedings, 1997, 127-134

[Gooch 2001] B. Gooch and A. Gooch, Non-Photorealistic Rendering, A K Peters, 2001

[Grabli 2002] S. Grabli, F.X. Sillion, S.R. Marschner, J.E. Lengyel, Image-Based Hair Capture by Inverse Lighting, Graphics Interface, 2002

[Hadap 2000] S. Hadap, and N. M. Thalmann, Interactive hair styler based on fluid flow, In Proceedings of Computer Animation and Simulation, 2000, 87-100

[Hadap 2001] S. Hadap, N. M. Thalmann. Modeling Dynamic Hair as a Continuum, Eurographics Proceedings, 2001, 20(3), 329-338

[Hausner 2001] A. Hausner. Simulating Decorative Mosaics, SIGGRAPH Proceedings, 2001, 573-580

[Henyey 1941] L. Henyey and J. Greenstein. Diffuse radiation in the galaxy, Astrophysics Journal, 93: 70-83,1941

[Jenkins 1976] F. Jenkins and H. White. Fundamentals of optics, McGraw-Hill, 1976

[Jensen 2001a] H. W. Jensen. Realistic Image Synthesis Using Photon Mapping, A. K. Peters, 2001

[Jensen 2001b] H. W. Jensen, S. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport, SIGGRAPH Proceedings, 2001

[Jones 2000] T. Jones and R. Perry, Antialiasing with line samples, MERL Technical report, June 2000, TR-2000-21

[Kajiya 1984] J. Kajiya and B. P. Herzen. Ray tracing volume densities, SIGGRAPH Proceedings, 1984, vol. 18, 165-174

[Kajiya 1989] J. Kajiya and T. Kay. Rendering fur with three dimensional textures, SIGGRAPH Proceedings, 1989, vol. 23, 1989, 271-280

[Keller 2001] A. Keller, Interweaved sampling, Eurographics Workshop on Rendering, 2001

[Kim 1994] M. Kim, E. Park, and H. Lee, Modeling and Animation of Generalized Cylinders with Variable Radius Offset Space Curves, The Journal of Visualization and Computer Animation, 5(4), 189-207

[Kim 2000] T. Kim and U.Neumann. A Thin shell volume for modeling human hair, IEEE Computer Animation, pp.121-128, 2000

[Kim 2001] T. Kim and U.Neumann. Opacity Shadow Maps, Eurographics Rendering Workshop, 2001

[Kim 2002] T. Kim and U.Neumann. Interactive Multiresolution Hair Modeling and Editing, SIGGRAPH Proceedings, 2002, 620-629

[Kishi 1998] K. Kishi and S. Morishima. Dynamic Modeling of Human Hair and GUI Based Hair Style Designing System, SIGGRAPH Technical Sketches, 1998, pp. 255

[Koh 2000] C. Koh, and Z. Huang. Real-Time Animation of Human Hair Modeled in Strips, Computer Animation and Simulation, 2000, 100-112

[Koh 2001] C. Koh, and Z. Huang. A simple physics model to animate human hair modeled in 2D strips in realtime, Computer Animation and Simulation, 2001, 127-138

[Kong 1998a] W. Kong and M. Nakajima. Generation of hair model from 2D image, The Journal of the Institute of Image Information and Television Engineers, Vol. 52, (9), 1998, 129-131

[Kong 1998b] W. Kong and M. Nakajima. Generation of 3D hair model from multiple images, The Journal of the Institute of Image Information and Television Engineers, Vol. 52, (9), 1998, 109-114

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

[Kong 2000] W. Kong and M. Nakajima. Hair rendering by jittering and pseudo shadow, Computer Graphics International, 2000, 287-291

[Kurihara 1993] T. Kurihara, K. Anjyo, and D. Thalmann, Hair Animation with Collision Detection, Proc. Computer Animation, 1993, 128-138

[LaFrance 2001] M. LaFrance, First Impressions and Hair Impressions, Unpublished manuscript, Yale University, Department of Psychology, New Haven, Connecticut, http://www.physique.com/sn/sn_yale_study2.asp

[LeBlanc 1991] A. LeBlanc, R. Turner and D. Thalmann. Rendering hair using pixel blending and shadow buffers, The Journal of Visualization and Computer Animation, vol. 2, 1991, 92-97

[Lee 2001] D.-W. Lee and H.-S. Ko, Natural Hairstyle Modeling and Animation, Graphical Models, 63(2), 67-85

[Levoy 1985] M. Levoy and T. Whitted, The Use of Points as a Display Primitive, Technical Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, January

[Levoy 1989] Marc Levoy. Display of surfaces from volume data, Ph.D. thesis, University of North Carolina at Chapel Hill, 1989.

[Lewis 2001] J. P. Lewis. Personal communications, 2001, Disney TSL.

[Lokovic 2000] T. Lokovic and E. Veach. Deep shadow maps, SIGGRAPH Proceedings, 2000, 385-392

[Luebke 2001] D. Luebke, and B. Hallen. Perceptually-Driven Simplification for Interactive Rendering, Rendering Techniques 2001, 221-232.

[Lengyel 2000] J. Lengyel, Realtime Fur, Rendering Techniques, 2000, 243-256

[Lengyel 2001] J. Lengyel, E. Praun, A. Finkelstein, H. Hoppe, Real-Time Fur over Arbitrary Surfaces, ACM Symposium on Interactive 3D Techniques, 2001, 227-232

[Lindholm 2001] E. Lindholm, M. Kligard and H. Moreton, A user-programmable vertex engine, SIGGRAPH Proceedings, 2001, 149-158

[Mausch 2000] S. Maush, A Fast Algorithm for Computing the Closest Point and Distance Transform, Submitted for publication in the Journal of SIAM SISC. http://www.acm.caltech.edu/~seanm/software/cpt/cpt.pdf

[McReynolds 1997] McReynolds (organizer). Programming with OpenGL: Advanced Techniques, SIGGRAPH Course Note, 1997

[Mitchell 1987] D. Mitchell, Generating antialiased images at low sampling densities, SIGGRAPH Proceedings, 1987, 65-72

[Mitchell 1996] D. Mitchell, Consequences of stratified sampling in graphics, SIGGRAPH Proceedings, 1996, 277-280

[Miller 1988] G. Miller, From Wire-Frame to Furry Animals, Graphics Interface, 1988, 138-146

[Möller 1999] T. Möller and E. Haines, Real-Time Rendering, A K Peters, 1999

[Newell 1972] M. Newell, R. Newell, and T.L. Sancha, A Solution to the Hidden Surface Problem, Proc. ACM National Conf., 1972

[Neyet 1995] F. Neyret. A general and multi-scale model for volumetric textures, Graphics Interface, 1995, 83-91

[Neyret 1998] F. Neyret, Modeling, animating, and rendering complex scenes using volumetric textures, IEEE Transactions on Visualization and Computer Graphics, 1998, 4(1), 55-70, 1998

[O'Brien 2001] D. O'Brien, S. Fisher, and M. Lin. Automatic Simplification of Particle System. Computer Animation Proceedings, 2001, 210-219

[Pai 2002] D. K. Pai, Strands: Interactive Simulation of Thin Solids using Cosserat Models. Eurographics Proceedings, 2002

[Parke 1996] F. Parke and K. Waters, Computer Facial Animation, A K Peters, 1996.

[Parraga 2000] C.A. Parraga, T. Troscianko, and D.J. Tolhurst, The human visual system is optimized for processing the spatial information in natural visual images, Current Biology, Vol 10, January 2000, 35-38

[Pattaniak 1998] S. Pattanaik, J. Ferwerda, M.D. Faichild, and D. P. Greenberg. A Multiscale Model of Adaptation and Spatial Vision for Realistic Image Display, SIGGRAPH Proceedings, 1998, 287-298

[Pellacini 2000] Pellacini, F., Ferwerda, J.A., and Greenberg.D.P. Toward a psychophysically-based light reflection model for image synthesis. SIGGRAPH Proceedings, 2000, 55-64.

[Perlin 1989] K. Perlin, Hypertexture, SIGGRAPH Proceedings, 1989, 253-262

[Plante 2001] E. Plante, M.-P. Cani, and P. Poulin,K. Perlin, A layered wisp model for simulating interactions inside long hair, Eurographics Computer Animation and Simulation 2001, Sep 2001, 139-148

[Provot 1995] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior, Graphics Interface, 1995, 147-154

[Rasmasubramanian 1999] Rasmasubramanian, M., Pattanaik, S. N., Greenberg, D. P., A Perceptually Based Physical Error Metric for Realistic Image Synthesis, Proceedings SIGGRAPH, 1999

[Reeves 1985] W. Reeves, Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particles Systems, SIGGRAPH Proceedings, 1985, 313-322

[Reeves 1987] W. T. Reeves, D. H. Salesin, and R. L. Cook, Rendering antialiased shadows with depth maps, SIGGRAPH Proceedings, vol. 21, 1987, 283-291

[Rockwood 1996] A. Rockwood and P. Chambers, Interactive Curves and Surfaces, Morgan Kaufmann Publishers Inc., San Francisco, 1996

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

[Robbins 1988] C. Robbins, Chemical and Physical Behavior of Human Hair, Springer-Verlag, 1988

[Shannon 1949] C. E. Shannon, Communication in the presence of noise, Proc. IRE Vol. 37, 1949, 10-21

[Singh 1998] K. Singh, and E. Fiume, Wires: A Geometric Deformation Technique, SIGGRAPH Proceedings, 1998, 405-415

[Smith 1995] A. R. Smith. A Pixel is Not A Little Square, A Pixel is Not A Little Square, A Pixel is Not A Little Square! (And a Voxel is Not a Little Cube), Tech Memo 6, Microsoft, Jul 1995

[Snyder 1998] J. Snyder, and J. Lengyel, Visibility sorting and compositing without splitting for image layer decomposition, SIGGRAPH Proceedings, 1998, 219-230

[Sourin 1996] A. Sourin, A. Pasko and V. Savchenko, Using Real Functions with Application to Hair Modeling, Computers and Graphics, vol. 20, (1), 1996, 11-19

[Stam 1995] J. Stam. Multi-Scale Stochastic Modeling of Complex Natural Phenomena, Ph.D. Thesis, Department of Computer Science, University of Toronto, 1995

[Stollnitz 1996] E. Stollnitz, T.D. Derose, and D.H. Salesin. Wavelets for Computer Graphics, Morgan Kaufmann

[Thalmann 1996] N. M. Thalmann, S. Carion, M. Courchesne, P. Volino, and Y.Wu, Virtual Clothes, Hair and Skin for Beautiful Top Models, Computer Graphics International, 1996, 132-141

[Thalmann 2000] N. M. Thalmann, S. Hadap, P. Kalra, State of the Art in Hair Simulation, International Workshop on Human Modeling and Animation, Seoul, Korea, June 2000, Korea Computer Graphics Society, 3-9.

[Turk 1991] G. Truk, Generating Textures on Arbitary Surface Using Reaction Diffusion, SIGGRAPH Proceedings, 1991, 21(4), 289-298

[Upstill 1992] S. Upstill, The Renderman Companion, Addison-Wesley publishing company, 1992

[Valkovic 1988] V. Valkovic, Human Hair, CRC Press Inc., 1988

[Volino 1995] P. Volino, M. Courchesne, and N. Thalmann. Versatile and efficient techniques for simulating cloth and other deformable object, SIGGRAPH Proceedings, 1995, 137-144

[Wang 1995] X. J. Wang, T.E. milner, R.P. Dhond, W. V. Sorin and S. A. Newton, Characterization of human scalp hair by optical low-coherence reflectometry, Optics Letters, March 1995, 20(6), 524-526

[Wang 2001] T. Wang and X.D. Yang. A design tool for the hierarchical hair model, Information Visualization, 2001. 186-191

[Watanabe 1992] Y. Watanabe and Y. Suenega. A trigonal prism-based method for hair image generation, IEEE Computer Graphics and Application, 12(1), January 1992, 47-53

[Westin 1992] S. Westin, J. Arvo, and K. Torrance, Predicting reflectance functions from complex surfaces, SIGGRAPH Proceedings, 1992, 255-264

[Whitted 1980] J. T. Whitted. An improved illumination model for shaded display, Comm. ACM, 1980, 23(6), 342-349

[Williams 1978] L. Williams. Casting curved shadows on curved surfaces, SIGGRAPH Proceedings, Sug 1979, vol. 12, 270-274.

[Woo 1990] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms, IEEE Computer Graphics and Applications, 10(6), November, 1990, 13-32

[Xu 2001] Z. Xu, and X.D. Yang, V-HairStudio: An Interactive Tool for Hair Design, IEEE Computer Graphics and Applications, May/June 2001, pp. 36-43.

[Yang 2000] X. D.Yang, Z. Xu, J. Yang, and T. Wang. The Cluster Hair Model, Graphical Models, 62, 2000, 85-103

[Yu 2001] Y. Yu, Modeling realistic virtual hairstyles, Pacific graphics 2001, 295-304