Modeling Expressive Wrinkles and Hair for Virtual Humans

by

Yosuke Bando

A Master Thesis

Submitted to the Graduate School of
Information Science and Technology
The University of Tokyo
on February 4, 2003
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science

Thesis Supervisor: Tomoyuki Nishita
Title: Professor of Computer Science

Abstract

As virtual humans appear more frequently in various fields such as movies, video games, commercial films, and virtual reality, it becomes increasingly important to display them realistically. However, expressive wrinkles and dynamic hair, which play crucial parts in synthesizing an impressive human head model, are generally difficult to model and animate interactively because of their complexity. This thesis presents methods for modeling wrinkles and hair over virtual human head models. Wrinkles are created by displacing the skin surface based on their specified locations and shapes. Wrinkle amplitude is dynamically modulated according to skin surface deformation while animating the face with facial muscles. Hair is modeled with particles, the sampling points of the volume of hair defined over the scalp of the head model, in order to avoid explicit simulation of a large number of hair strands. Complex interactions of hair with the head, the hair itself, and the environment such as wind are rapidly handled with the particle dynamics. Both wrinkles and hair can be animated interactively.

Acknowledgements

First of all, I would like to thank Professor Nishita for his useful advice and discussion. I also would like to thank the members of ATR Human Information Science Laboratories Department 2 for providing materials and a work environment for the work on modeling wrinkles. I acknowledge the BRDF data of human skin from the Cornell University Program of Computer Graphics. Finally, I would like to express my gratitude to all of the members in Nishita Lab.

Contents

Chapter 1

Introduction

In the field of Computer Graphics (CG), one of the most difficult things is to display humans. Nowadays CG has proven to be superior in modeling and displaying rigid objects such as buildings and machinery, but it still has some difficulties dealing with soft, fuzzy, or vague things. Human is one of the examples that have these properties.

Recently, virtual humans appear frequently in various fields such as movies, video games, commercial films and virtual reality. And it becomes increasingly important to display them realistically. For this reason, many researches have been undertaken in various aspects of displaying virtual humans: modeling, rendering and animation of human body, skin, hair, clothes, etc. This thesis addresses problems concerning two of them, skin and hair, which play crucial roles in the synthesis of impressive human models, especially for human head models.

For skin, we focus on its geometric details, i.e. wrinkles, particularly for a face. Realism of rendered human skin can be strongly enhanced by taking into account skin wrinkles. However, modeling wrinkles is a difficult task, and considerable time and effort are necessary to achieve satisfactory results. Chapter 2 presents a simple method for modeling expressive wrinkles on the face, nevertheless taking into account the properties of real wrinkles. Wrinkles are specified using intuitive parameters, and the wrinkled face is rendered at an interactive frame rate, dynamically modulating wrinkle amplitude according to skin surface deformation while animating the face with facial muscles. We demonstrate the ability of our method to model realistic wrinkle shapes by comparing them with real wrinkles.

Hair is also an obstacle to the production of human models because the enormous number of hair strands and their thin nature complicate the

situation. Chapter 3 presents a practical method for modeling hair that takes into account the interactions among hair. We model the hair as a set of particles that serve as sampling points of the volume of the hair, which covers the whole region where hair is present. Then, the dynamics of the hair is simulated using the interacting particles. The novelty of this approach is that as opposed to the traditional way of modeling hair, we release the particles from tight structures that are usually used to represent hair strands or clusters. Therefore, by making the connections between the particles loose while maintaining their overall stiffness, the hair can be dynamically split and merged during lateral motion without losing its lengthwise coherence.

Chapter 2 describes modeling of expressive wrinkles and Chapter 3 describes modeling of hair with corresponding results and discussions. A detailed introduction and a review of the related work for each topic is also presented in each chapter. Finally, conclusions for this whole thesis are given in Chapter 4.

Chapter 2

Modeling of Expressive Wrinkles

This chapter presents a simple method for modeling wrinkles on human skin. The main focus is on the expressive wrinkles on the face, but our method is also applicable to the wrinkles on the other parts of the body.

2.1 Introduction

Wrinkling, together with skin color, is the main factor which determines the realism of skin rendering. Traditionally, however, rendering skin with wrinkles involves painstaking tasks such as manually drawing textures, and considerable time and effort are necessary to achieve satisfactory results. Our goal is to reduce this burden with a new simple method for modeling wrinkles.

Wrinkles can roughly be classified into two types: *fine-scale wrinkles* and *large-scale wrinkles* (see Section 2.3). Expressive wrinkles belong to large-scale wrinkles, and we model them by deforming the facial mesh based on the specified location and shape of each wrinkle. In addition, we create fine-scale wrinkles to add details to the skin surface in order to further enhance the realism of skin rendering. We model them by carving their furrows along a user-specified direction field over the mesh.

As this is a simple top-down approach, users can specify wrinkles using intuitive parameters (e.g., direction, depth and width) and easily model wrinkles as they desire. Thanks to their simplicity, the wrinkle generation algorithms are easy to implement and perform relatively fast. Wrinkled skin surfaces are rendered at an interactive frame rate, dynamically modulating wrinkle amplitude according to skin surface deformation during animation.

Nevertheless, the created wrinkles satisfactorily enhance the realism of skin rendering since the properties of real wrinkles are taken into account. We demonstrate the ability of our method to model realistic shapes of expressive wrinkles by comparing them with real wrinkles.

In the following sections we describe the process in detail. Although our main focus is on modeling expressive wrinles, we do not confine ourselves to dealing only with the facial mesh. Section 2.2 presents a review of related work. Section 2.3 explains the properties of wrinkles on human skin that our method takes into account. Section 2.4 describes how we construct the body-part mesh so that the wrinkle modeling method can be applied to it. Sections 2.5 describes modeling of large-scale wrinkles, which serve as expressive wrinkles when created over the face. Section 2.6 describes modeling of fine-scale wrinkles for further realism. Section 2.7 shows results and Section 2.8 discusses the advantages and the limitaions of our method.

2.2 Related Work

Some studies modeled the skin as several layers, and animated the face by physically-based simulation [35, 42]. As the result of skin deformation, some kinds of wrinkles were formed. Boissieux et al. [4] simulated various kinds of wrinkles with a more sophisticated method, although on an abstract, simplified piece of skin. They also proposed an image-based approach in the same paper. Viaud and Yahia [37] modeled wrinkle bulges as spline segments. Wu et al. [43, 44, 45] presented a method to simulate facial animation with both fine- and large-scale wrinkles (one of these also dealt with static wrinkles on the hand [43]). Volino and Thalmann [39] rapidly animated wrinkles on deformable models by modulating the amplitude of a given wrinkle pattern. This work was further extended mainly for cloth wrinkling by Hadap et al. [15].

For creating fine-scale wrinkles, only a few methods have been proposed. Ishii et al. [18] divided skin surface into polygons by hierarchical Voronoi division, and rendered each edge of these polygons as a furrow of a wrinkle. Wu et al. [43] employed a hierarchical Delauney triangulation instead. Other studies captured and reconstructed details of the skin surface from actual skin samples by using a laser range scanner [30] and by image analysis [17].

Our approach differs from these methods principally in that it provides easy control over wrinkle characteristics. In contrast, physically-based simulation does not facilitate direct manipulation of wrinkles. Moreover, it is

difficult for users to modify the wrinkle patterns acquired from actual skin samples or images. The works by Volino and Thalmann, and Hadap et al. assumed wrinkle pattern textures to be drawn by the user, although Hadap et al. provided strain patterns in a garment as a guide. We demonstrate the ability of our method to model realistic shapes of expressive wrinkles by comparing them with real wrinkles, whereas none of the above works for modeling wrinkles are guaranteed to have such abilities. Another difference is that our method dynamically modulates the amplitude of both fine-scale and large-scale wrinkles according to the skin surface deformation during animation.

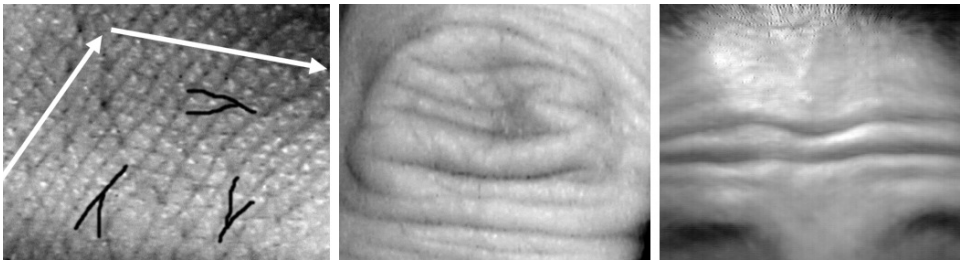
2.3 Wrinkle Properties

Wrinkles are the main factor determining the undulation of the skin surface. We can roughly distinguish two types of wrinkle and we define them as follows: fine-scale wrinkles are small wrinkles that cover the entire skin surface; large-scale wrinkles are distinct wrinkles unique to a particular body part (e.g., the expressive wrinkles on the face and the wrinkles on the finger joints).

Fine-scale wrinkles have a similar structure over the entire body except palms and soles in the sense that their furrows intersect each other over the surface [28]. These furrows run in particular directions depending on the location on the skin. As shown in Figure 2.1(a), a close look at the skin surface reveals their properties to be the following. 1) Furrows run locally in two directions. 2) The intersection point of two furrows running in the same direction becomes the endpoint of either. The first property has some exceptions, but holds over most of the surface. This is also supported by the observation that the outline of intersecting furrows is roughly diamond-shaped [29].

Large-scale wrinkles are formed due to shrinking of the skin surface caused by body part movement. Skin material itself is incompressible [12], and therefore, when the skin surface shrinks, the excess skin buckles and forms wrinkles. These become gradually more prominent with repeated wrinkling and age. As shown in Figures 2.1(b) and 2.1(c), we can observe that large-scale wrinkles have the following property: each has a sharp furrow at its center line and round bulges on its both sides.

Both types of wrinkle become more pronounced when the skin surface shrinks perpendicularly to their directions.



(a) (b) (c)

Figure 2.1: Photographs of wrinkles. (a) Fine-scale wrinkles on the back of the hand. White arrows show two wrinkle directions. Black lines show some of the intersection points of wrinkle furrows. (b) Large-scale wrinkles around the finger joint. (c) Large-scale wrinkles on the forehead.

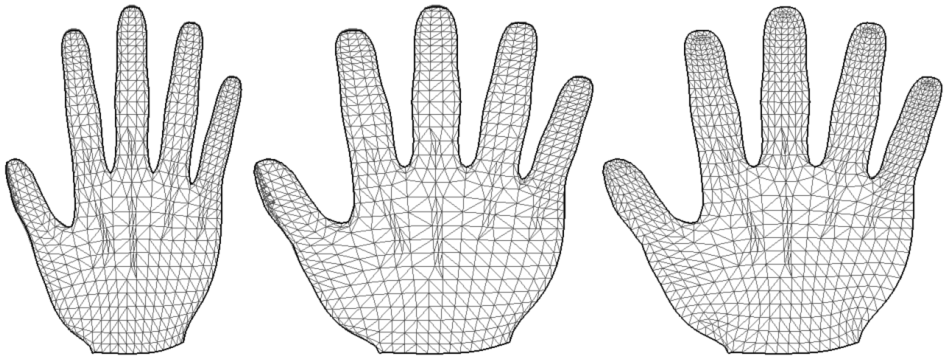
2.4 Construction of Body-part Mesh

We use a triangle mesh to represent a face and other body parts. This section describes how to construct a mesh so that the wrinkle modeling methods described in the following sections can be applied to it.

The geometry of the mesh is modeled with CG software, or acquired with a range scanner. The resolution of the mesh can be arbitrary. If the mesh is not fine enough to represent large-scale wrinkles, we apply the adaptive refinement proposed by Volino and Thalmann [39].

The mesh is mapped to the two-dimensional texture space in order to make the wrinkle modeling process simpler and faster (see Sections 2.5 and 2.6). The mapping should be such that two distinct points on the mesh do not correspond to a single point in the texture space (one to one mapping), and that the topology of the mapped mesh is the same as that of the original mesh (topology preserving). We use planar or cylindrical projection depending on the shape of the body part. Cylindrical projection may produce a seam in the mapped mesh, but this will not be a problem if we treat the texture space as wrapped around. If distortion of the current projected mesh is not negligible, we fix the boundary vertices of the mesh, and smooth the internal vertices by minimizing the total energy of springs placed along the edges. We set spring constants in the same way as Eck et al. [13] constructed harmonic maps. In our case, however, the boundary is not generally convex as opposed to the requirement for harmonic maps, but this method works fine if we move some vertices so that the boundary becomes less concave. Figure 2.2 shows this distortion minimization process

for the hand mesh. The mesh is scaled as shown in Figure 2.2(b) so that it better fits the unit square texture space. We did not apply this process to the facial mesh since we obtained an acceptable mapping with cylindrical projection alone.



(a) (b) (c)

Figure 2.2: (a) The hand mesh mapped with planar projection. (b) The mesh modified by scaling and by moving some boundary vertices. (c) The mesh after distortion minimization.

2.5 Modeling of Large-scale Wrinkles

As described in Section 2.3, large-scale wrinkles are distinct, local wrinkles formed due to shrinking of the skin surface. Therefore, we have the user specify wrinkles one by one, by their locations and shapes. The body-part mesh is then deformed based on this information to represent the wrinkles.

2.5.1 Wrinkle Location and Shape

Location of a wrinkle is specified in the texture space by drawing a cubic Bezier curve as a furrow of the wrinkle as shown in Figure 2.4(b). The cross sectional shape of the wrinkle is given as *wrinkle shape function* $S(l)$, a height function of distance l from the specified furrow. l is measured in perpendicular direction to the furrow in the texture space as shown in Figure 2.3(a). We adopt Equation 2.1 as the wrinkle shape function for the following three reasons.

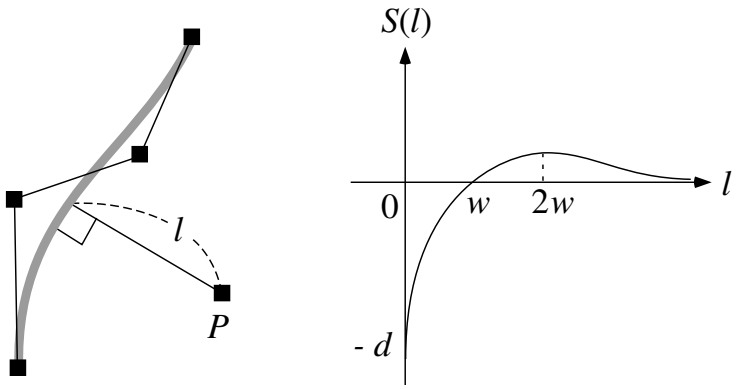
$$S(l) = d(l/w - 1)\exp(-l/w). \tag{2.1}$$

First, as shown in Figure 2.3(b), it simulates the wrinkle property that the furrow of a wrinkle is sharp and the bulges round (see Section 2.3). Second, it satisfies the following equation.

$$\int_0^\infty S(l)dl = 0. \tag{2.2}$$

This means that the volume depressed at the furrow goes to the side to make up the bulge, which simulates the incompressibility of skin. Third, it is easily controlled by two intuitive parameters, depth d and width w.

To represent wrinkles, the body-part mesh is deformed by displacing its vertices along their normal vectors according to the wrinkle shape function. If the mesh is not fine enough to represent wrinkles, we apply the adaptive refinement [39] around the specified wrinkle locations. Displacing only vertices near the wrinkle locations improves computational efficiency, and wrinkles are generated on the fly. Therefore, users can see what happens immediately after they modify the parameters assigned to each wrinkle.



(a) (b)

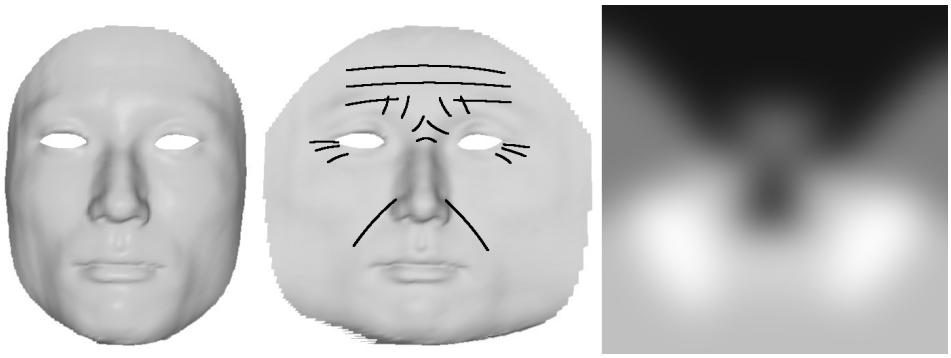
Figure 2.3: (a) A cubic Bezier curve representing the location of a wrinkle furrow in the texture space. Distance from the furrow is measured in its perpendicular direction. (b) The wrinkle shape function.

2.5.2 Animating Body-part Mesh

To simulate the wrinkles changing their shapes according to body part movement, the way of animating the body-part mesh is defined. In order to display the skin with wrinkles at an interactive frame rate, we need a relatively simple and fast method, and we use the facial muscle model proposed by Waters [40] to animate the facial mesh. Based on this model, we add the

effect of thickness of the subcutaneous tissue (the tissue under the skin surface) to generate more expressive facial animation. Other body parts are animated with skeletons [38].

The facial muscle model of Waters generates a displacement field which indicates how far each point of the facial skin is displaced toward the muscle origin (the point from which the muscle originates) as a result of muscle contraction. Waters modeled this displacement field with a combination of simple analytic functions [40]. This method works fast, but the contraction of the subcutaneous tissue was not considered. Since the skin is almost incompressible [12], the excess tissue should lift the skin surface up in the outward direction. To simulate this, we compute the area expansion ratio of the skin surface. We first compute the ratio of the current area of each triangle of the mesh to its initial area, and obtain the expansion ratio e_i at vertex i of the mesh by averaging the values of triangles to which vertex i belongs. Then, if the initial thickness of the subcutaneous tissue under vertex i is h_i^0, the current thickness h_i is h_i^0/e_i, assuming the volume preservation. Thus we displace the vertex along its normal vector by $h_i^0/e_i - h_i^0$. The initial thickness of the subcutaneous tissue is given as a grayscale image in the texture space as shown in Figure 2.4(c).



(a) (b) (c)

Figure 2.4: (a) The facial mesh. (b) The wrinkle locaitons specified on the mesh in the texture space. (c) The initial thickness of the subcutaneous tissue.

2.5.3 Modulation of Wrinkle Amplitude

While animating the body part, wrinkles are also animated by dynamically modulating their amplitude according to skin surface deformation. To mea-

sure the deformation, we compute shrinkage s of the skin surface at each vertex of the body-part mesh. Shrinkage inside the triangle of the mesh is interpolated from its three vertices. Then we modulate wrinkle amplitude by the *amplitude modulation function* $M(s)$.

The following explains how to compute skin shrinkage of the animated mesh. As described in Section 2.3, wrinkles become more pronounced when the skin surface shrinks perpendicularly to their directions. The direction of a large-scale wrinkle is given as the tangential vector of the cubic Bezier curve indicating the wrinkle location, and shrinkage is measured in its perpendicular direction, $\mathbf{d}_p$, in the texture space. Figure 2.5(a) shows the vertex of the mesh (denoted by v_1) in the texture space for which we want to compute shrinkage, and we consider triangle $v_1v_2v_3$ of the mesh located in direction $\mathbf{d}_p$ from v_1. The original length L of the triangle is defined as the distance between vertex v_1 and edge v_2v_3 in direction $\mathbf{d}_p$ as shown in Figure 2.5(a). Suppose that the length of edge v_1v_2 and that of edge v_1v_3 are scaled by t_2 and t_3, respectively in the three-dimensional object space as the result of animating the body-part mesh. Assuming that the lengths of these edges are scaled by the same values in the texture space, the current length L' of the triangle is defined similarly as shown in Figure 2.5(b) (however, we do not actually change the texture coordinates of vertices v_2 and v_3). From these values, we compute shrinkage s as follows.

$$s = \frac{L - L'}{L} = 1 - \frac{L'}{L}. \tag{2.3}$$

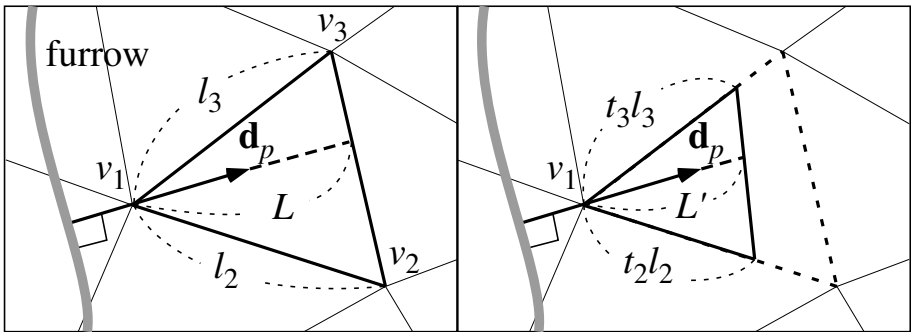
s is the amount of shortening relative to the original length, which equals 0 in the rest state (i.e., the state in which the mesh is not deformed). s becomes negative when the skin expands, and increases up to the maximum value of 1 as the skin shrinks. Generally, there are two triangles located in the direction perpendicular to the wrinkle furrow, in direction $\mathbf{d}_p$ and $-\mathbf{d}_p$ from v_1. In this case, we average two shrinkage values. Note that shrinkage s is dependent on wrinkle directions, and varies for different wrinkles even if it is measured at the same location. This realizes anisotropic wrinkle response to skin surface deformation.

According to the shrinkage s, wrinkle amplitude (or height) is scaled by the amplitude modulation function $M(s)$ defined as Equation 2.4. Figure 2.6 shows its graph. It is C^1 continuous at $s = 0$. Parameter r, in the range $[0, 1]$, controls the initial height of a wrinkle relative to its maximum height. Wrinkles become more pronounced when the skin surface shrinks, and less when it expands. If $r = 0$, $M(s) = 0$ for $s < 0$, and wrinkles do not appear

unless the skin surface shrinks.

$$M(s) = \begin{cases} (1-r)s + r & s \geq 0 \\ r\exp\{(1/r-1)s\} & s < 0 \end{cases} . \tag{2.4}$$

Parameter r is assigned to each wrinkle. The vertices relevant to a particular wrinkle are displaced according to the wrinkle shape function and the amplitude modulation function with the parameters assigned to this wrinkle.



(a) (b)

Figure 2.5: (a) The original length of a triangle measured in the texture space perpendicularly to the wrinkle direction. (b) The current length of the triangle. The lengths of two edges in the texture space are assumed to be scaled by the same values as in the three-dimensional object space.

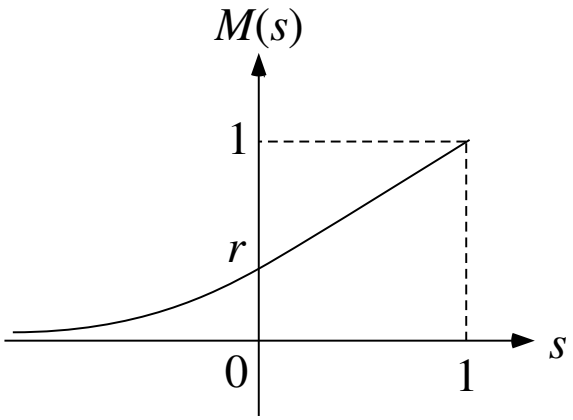


Figure 2.6: The amplitude modulation function.

2.6 Modeling of Fine-scale Wrinkles

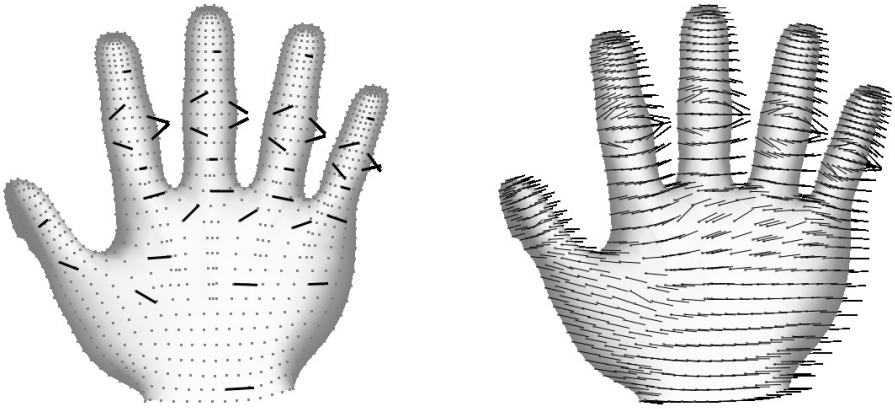
In addition to model large-scale wrinkles that serve as expressive wrinkles when created on the face, we also model fine-scale wrinkles to add details to the skin surface in order to further enhance the realism of skin rendering. As described in Section 2.3, fine-scale wrinkles cover the entire skin surface, and are similarly structured in the sense that their furrows run along particular directions depending on their location on the skin. To simulate these characteristics, wrinkle furrows are generated along a direction field over the body-part mesh. The user first specifies a direction field in the texture space in order to indicate the wrinkle directions. Next, wrinkle furrows are carved along the direction field, and the resultant undulation of the skin surface is recorded as a height field. Wrinkle characteristics are controlled by the magnitude of the direction field. Following the wrinkle property that the furrows of fine-scale wrinkles run locally in two directions (see Section 2.3), this procedure is performed independently for the two different direction fields, and two height fields are recorded as shown in Figure 2.9. These are superimposed, and the skin with fine-scale wrinkles is displayed by bump-mapping [3].

2.6.1 Formation of Direction Field

The direction field is used to indicate the directions of fine-scale wrinkles. It is formed in the texture space, and is represented as a direction vector at each vertex of the mapped mesh. The user first specifies direction vectors at several vertices as shown in Figure 2.7(a). Note that the mesh is mapped to the texture space, but it is shaded based on its geometry to inform the user of its shape. Next, direction vectors at the rest of the vertices are interpolated from these vectors. For interpolation, again we minimize the spring energy similarly to the mapping distortion minimization described in Section 2.4. In this case, however, the values associated to the vertices are direction vectors instead of texture coordinates, and we fix the user-specified direction vectors instead of the texture coordinates of the boundary vertices. The resultant vectors form a direction field as shown in Figure 2.7(b).

2.6.2 Carving of Wrinkle Furrows

Wrinkles are generated by carving their furrows along the direction field. Since we work in the texture space, the undulation of the skin surface due to these furrows is represented as a height field. This height field is defined



(a) (b)

Figure 2.7: (a) The direction vectors specified at several vertices of the hand mesh. (b) The direction field formed after interpolation.

as *height image*, a grayscale image whose intensity represents the height value. This image is initialized in white, and wrinkle furrows are carved by drawing gray line segments on it. In addition, we use another image called *occupancy image* to control the distance between adjacent furrows. This is a binary image which records the regions occupied by the already generated furrows. We control the distance between furrows by forbidding new furrows to be generated from these regions. The occupancy image is initialized to 0 (not forbidden) where the mesh is mapped, and 1 (forbidden) otherwise, because we do not want to generate wrinkles that have no correspondence to the mesh. As most graphics systems have double buffers for animation, we use one for the height image, and the other for the occupancy image.

A single wrinkle furrow is generated from a given point P in the texture space by the algorithm in the following sequence of steps (also see Figure 2.8). Note that each vertex has a direction vector which indicates the wrinkle direction. n holds the number of furrow segments to generate, and is initialized to infinity. The way to determine the parameters of wrinkle, N, θ, D, I, W_h and W_o, is described in Section 2.6.3.

1. Obtain wrinkle direction $\mathbf{d}$ at point P by interpolating from three vertices of the triangle to which P belongs.

2. Compute N, the number of furrow segments to generate from point P. Set n to $\min(n, N)$.

3. Add perturbation to wrinkle direction $\mathbf{d}$ by rotating it by angle θ.

4. Let Q be the point at distance D from point P in direction $\mathbf{d}$. If segment PQ intersects other wrinkle furrows, reset Q to the intersection point closest to P, and set n to 0.

5. Carve one segment of the wrinkle furrow by drawing line segment PQ on the height image with intensity I and width W_h.

6. Record the region occupied by the furrow by drawing line segment PQ on the occupancy image with value (or 'intensity') 1 and width W_o.

7. Stop if $n = 0$. Otherwise reset P to Q, decrement n by 1, and return to step 1.

We also apply this algorithm to the opposite direction field, setting $\mathbf{d}$ to $-\mathbf{d}$. Therefore, a wrinkle furrow grows to both sides of the starting point. In step 2, the number of furrow segments that can be generated from the current point is computed. If this number (denoted by N) is smaller than n, then n is reset to N. This determines how far the furrow grows depending on its location. In step 3, perturbation is added to the wrinkle direction to generate a natural furrow which otherwise becomes regular. In step 4, one furrow segment is determined. Distance D corresponds to its length. The latter half of step 4 follows one of the wrinkle properties described in Section 2.3: the intersection point of two furrows becomes the endpoint of either. If the current furrow intersects another furrow that has already been generated, it terminates its growth there as shown in Figure 2.8(b-2). In step 5, one furrow segment is carved. Intensity I and width W_h correspond to its depth and width, respectively. In step 6, the region occupied by the generated furrow segment is recorded so that no other furrow is generated from there (since the occupancy image is binary, the region is recorded as value 1). Thus a larger W_o keeps other furrows further away from the current one. In step 7, if the number of furrow segments to generate is 0, the algorithm terminates. Otherwise P is set to point Q, the current endpoint of the furrow, as a new starting point from which the furrow grows. n is decremented by 1 because one segment was generated, and we return to step 1 to generate more segments.

Using the above algorithm, wrinkle furrows are generated from a sequence of starting points until they cover the entire mesh in the texture space. We randomly select point P on the mapped mesh, and generate a

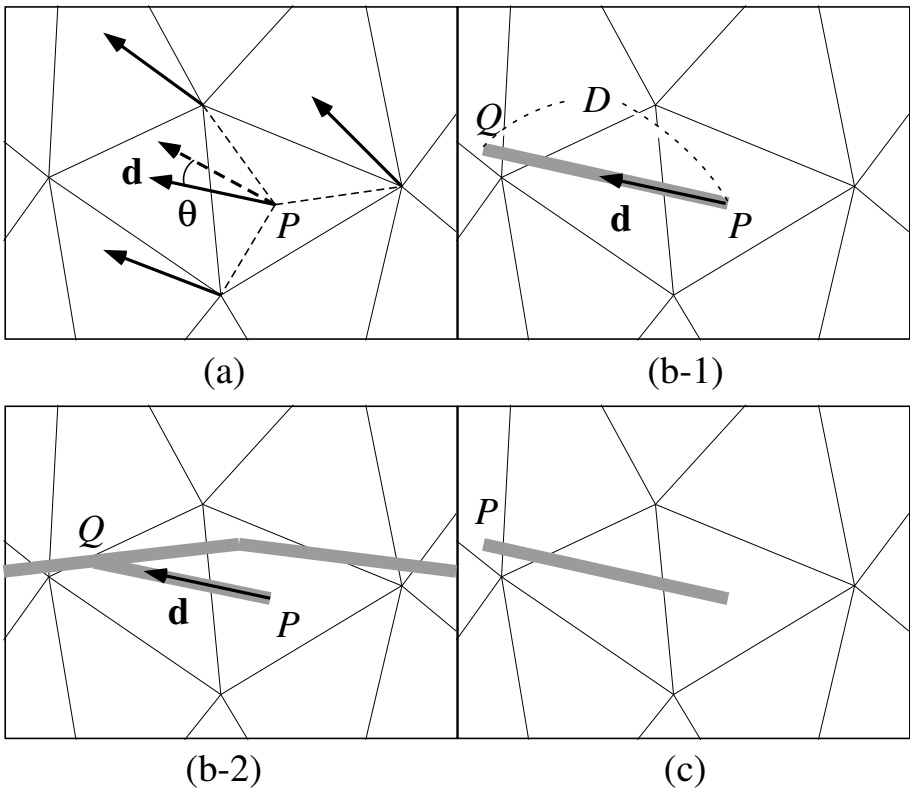
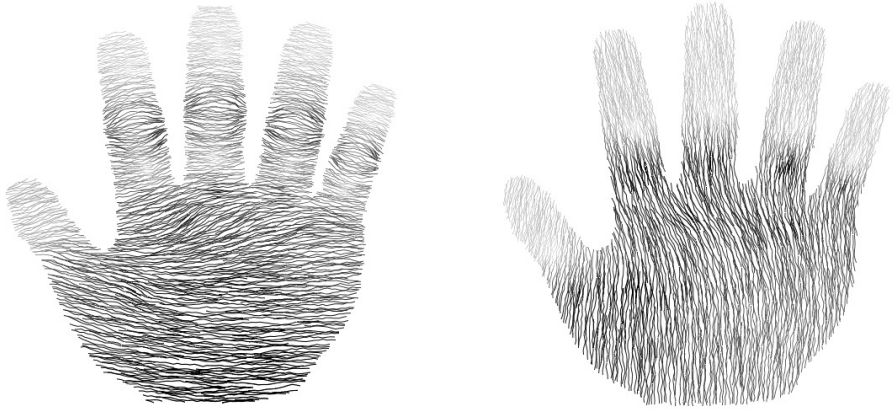


Figure 2.8: (a) Steps 1-3. The wrinkle direction obtained by interpolation and perturbation. (b) Steps 4-6. Generated one segment of the wrinkle furrow. (b-1) With no intersection with other furrows. (b-2) With intersection. (c) Step 7. A new starting point of the furrow preceded by the case shown in (b-1).

furrow from P if the value of the occupancy image at P is 0. This process is iterated until many furrows fail to be generated. Now that the mesh is roughly covered with wrinkles, we fill in the rest by generating furrows from each point in the texture space whose value is 0 in the occupancy image. The way of selecting starting points does not make much difference in the final result as long as the point distribution is relatively uniform. Figure 2.9(a) shows the resultant height field for the direction field shown in Figure 2.7(b).



(a) (b)

Figure 2.9: Two height fields over the hand. Each represents the undulation of the skin surface due to the wrinkle furrows generated along two different direction fields.

2.6.3 Controlling Wrinkle Characteristics

We introduced six parameters that control wrinkle characteristics in the generation algorithm described above. N: the number of furrow segments, θ: perturbation angle for the wrinkle direction, D: length of the furrow segment, I: depth of the furrow, W_h: width of the furrow, and W_o: distance between adjacent furrows. All these values are determined according to the magnitude of the direction field. If the user specifies the direction vector with larger magnitude at a particular vertex, wrinkle furrows around the area get more strongly oriented. That is, they become longer (having many long segments), deeper and wider, with smaller irregularity (smaller perturbation) and with larger distance to adjacent furrows. In our empirical model, Equation 2.5 defines these six parameters as functions of magnitude m of the direction vector at the point from which the furrow grows. m is

normalized to be in the range $[0, 1]$.

$$\begin{array}{rcl} N & = & \lfloor 30 - 20m \rfloor \\ \theta & = & (0.5 - 0.4m)r \\ D & = & 4 + 2m \\ I & = & 1 - m^2 \\ W_h & = & 2m \\ W_0 & = & 1 + 12m \end{array} , \tag{2.5}$$

where r is a random number in the range $(-1, 1)$. θ is expressed in radians. The unit length for D, W_h and W_o is usually set to the size of a pixel in the height image and the occupancy image. However, this depends on the size of the images and the body part, and only the relative values are important. The average magnitude of the direction vectors shown in Figure 2.7(a) is about 0.5.

2.6.4 Other Issues

Modeling wrinkles over multiple body-parts. It is not practical to generate wrinkles over a whole body mesh at once because this requires enormous texture size and also makes it difficult to map the mesh to the texture space with small distortion. Therefore, the mesh is divided into several body parts, and fine-scale wrinkles are generated over each of them. In order to generate seamless height fields between two neighboring body parts, the divided body-part meshes share their boundary portion. After wrinkles are generated over one body part, the direction field and the height field over the shared portion of the mesh are copied to the other body-part mesh. Starting with this initial state, wrinkles are generated over the other body part.

Animating fine-scale wrinkles. We can apply the amplitude modulation described in Section 2.5.3 to fine-scale wrinkles. At each vertex of the mesh, we first obtain the direction of fine-scale wrinkles from the direction field and compute the shrinkage of skin in its perpendicular direction. Then we interpolate the shrikage values inside the triangles from their vertices to form the shrinkage field in the texture space. Finally, we modulate the height field using the amplitude modulation function. Parameter r in Equation 2.4 is set to non-zero value (usually 0.5) everywhere in the texture space since the wrinkles are already visible in the rest state. Since we have two height field textures for the two different direction fields, amplitude modulation is

performed for each texture with its corresponding direction field, and the two modulated textures are superimposed and bump-mapped.

2.7 Results

Figures 2.10, 2.11, 2.12, and 2.13 show examples of the face with and without wrinkles. Large-scale wrinkles serve as expressive wrinkles on the face, making the face more expressive. The facial mesh was acquired by scanning a subject's face with neutral expression using a color 3D scanner 3030RGB/PS (Cyberware). It consists of about 20,000 triangles, and is fine enough to represent large-scale wrinkles. The values of the shape parameters for d and w for these wrinkles range from 2 mm to 12 mm, and parameter r is set to 0 so that the wrinkles do not appear at neutral expression. In addition, these figures show examples of various skin colors. Skin colors in Figures 2.10 and 2.11 were rendered taking into account the amount of skin pigment. Possible skin colors lie on a two-dimensional surface patch within RGB color space, with two axes corresponding to the amount of melanin and hemoglobin [9]. We simplified the surface patch to be planar, and determined skin color $\mathbf{c}$ by the amount of melanin m and that of hemoglobin h as follows.

$$\mathbf{c} = \mathbf{b} - m\mathbf{a}_m - h\mathbf{a}_h, \tag{2.6}$$

where $\mathbf{b}$ is the base color of skin, and $\mathbf{a}_m$ and $\mathbf{a}_h$ are the light absorption ratios of melanin and hemoglobin, respectively. Their values were determined empirically. We used these colors as ambient and diffuse reflectivity for rendering. The color in Figure 2.11 is due to large amount of melanin. Colors in Figures 2.12 and 2.13 were rendered by the Lafortune model [24] with the BRDF (*bi-directional reflectance distribution function*) data of skin measured by Marschner et al. [26].

We demonstrate the ability of our method to model realistic wrinkle shapes by comparing them with real wrinkles. By using a 3D digitizer VIVID700 (Minolta Co.), we digitized real skin surfaces of a forehead when it was wrinkled (Figure 2.15(a)) and when it was not (Figure 2.14). We specified large-scale wrinkles on the mesh shown in Figure 2.14, and wrinkles were generated after animating the forehead mesh to shrink the skin surface. Figure 2.15(b) shows the resultant mesh. We compared the two meshes shown in Figure 2.15 by measuring the distance from each vertex of one mesh to the nearest triangle of the other mesh. The average distance was 0.5 mm, which is relatively small in comparison to the depth of forehead

wrinkles (about 5 mm).

In addition to the expressive wrinkles, we created wrinkles for several other body-parts. Figure 2.16 shows examples of the hand with and without wrinkles. The original mesh consists of about 4,000 triangles, and we applied the adaptive refinement in order to represent the large-scale wrinkles specified around the finger joints. As a result, about 34,000 triangles were rendered. The parameter values for d and w range from 0.5 mm to 1 mm. We set parameter r to 1 for all of them because the fingers of the hand mesh are stretched at the rest state and the wrinkles should be at their maximum height. Figure 2.17 shows the simulated skin surface of the animated hand. Bump-mapping is enhanced excessively so that fine-scale wrinkles are clearly visible. The shapes of wrinkles running in the direction of skin deformation remain almost the same, whereas those of wrinkles running perpendicularly become less pronounced as the skin surface expands. Figure 2.18 shows the examples of the foot and Figure 2.19 shows the hand with the arm. These images were rendered with other objects and shadows at the cost of rendering time. Figure 2.20 shows the aged skin on the hand and the face with prominent wrinkles in the rest state.

Our method generates and renders wrinkles relatively fast. Large-scale wrinkles are generated on the fly. Therefore, users can see what happens immediately after they modify the wrinkle parameters. Fine-scale wrinkles are generated in about 10 seconds. And a body-part mesh with fine-scale and large-scale wrinkles is animated and rendered at an interactive frame rate. The face is displayed at 5.0 fps, and the hand at 6.4 fps on a Xeon 2 GHz with a Wildcat II 5110 graphics card. Image size and texture size are 512×512.

2.8 Discussion

In this chapter we have presented a simple method to easily model wrinkles on human skin. Large-scale wrinkles serve as expressive wrinkles when created on the face, and fine-scale wrinkles add details to the skin surface for further realism. The proposed method can also be applied to the other body parts. Its advantages are the following:

- It provides easy control over wrinkle characteristics. Users can specify wrinkles using intuitive parameters, and easily model wrinkles as they desire.

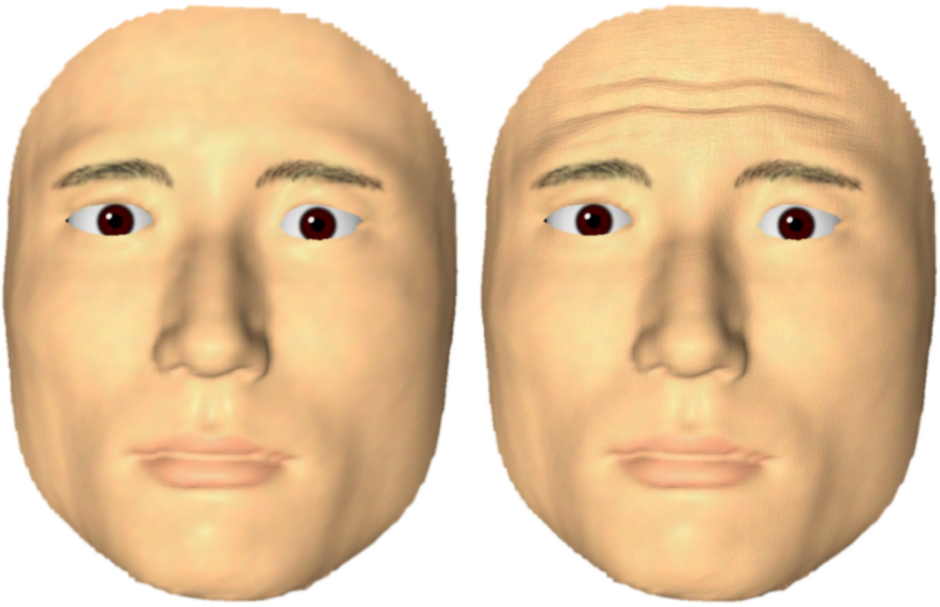


Figure 2.10: The thinking faces with and without wrinkles.

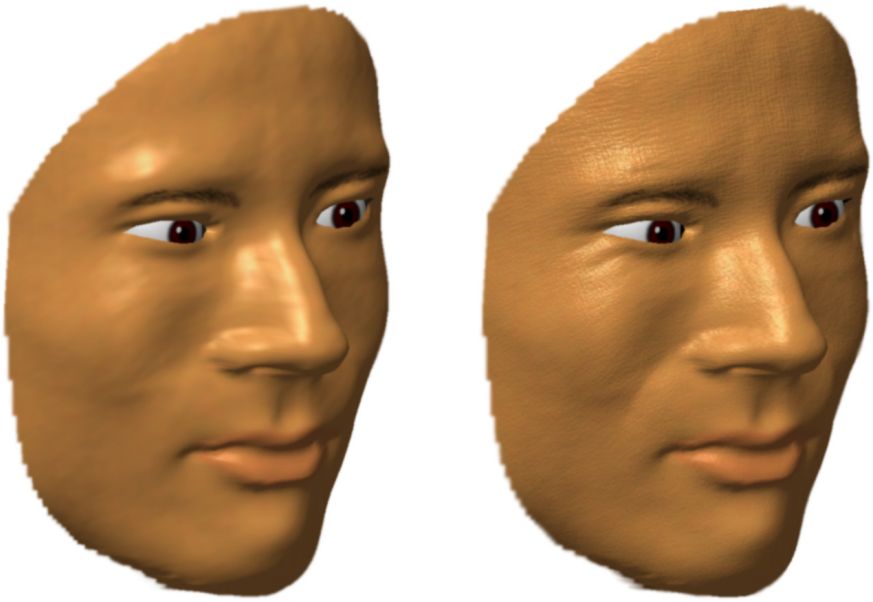


Figure 2.11: The smiling faces with and without wrinkles.

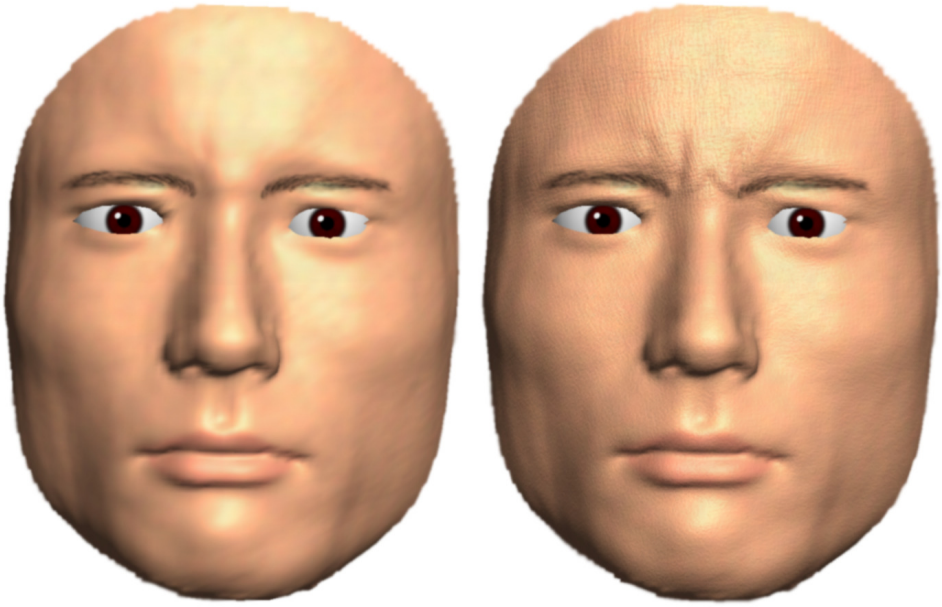


Figure 2.12: The troubled faces with and without wrinkles.

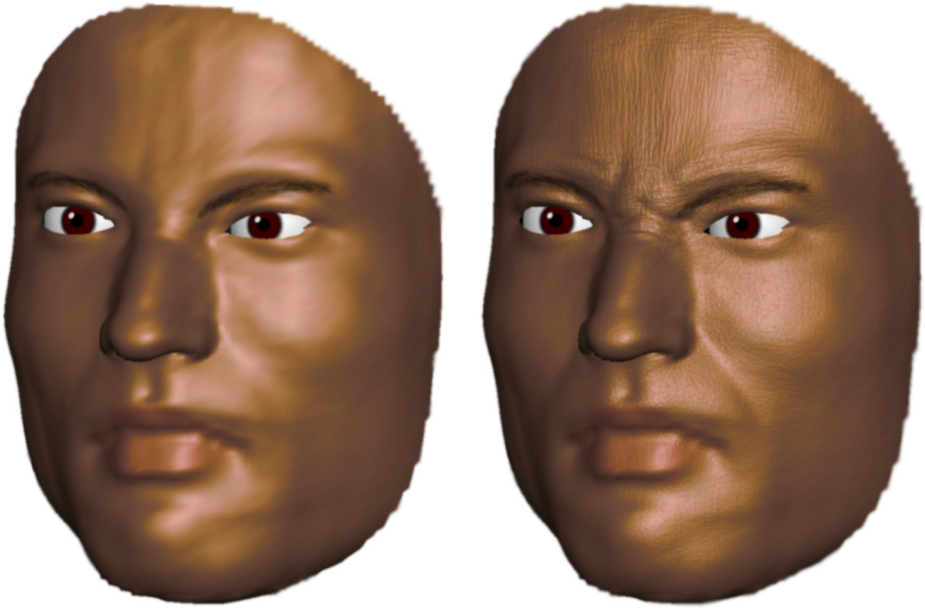


Figure 2.13: The angry faces with and without wrinkles.

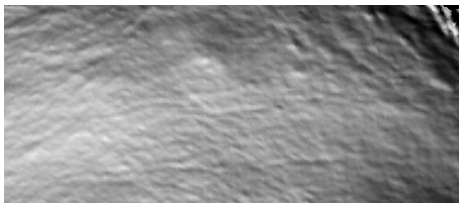
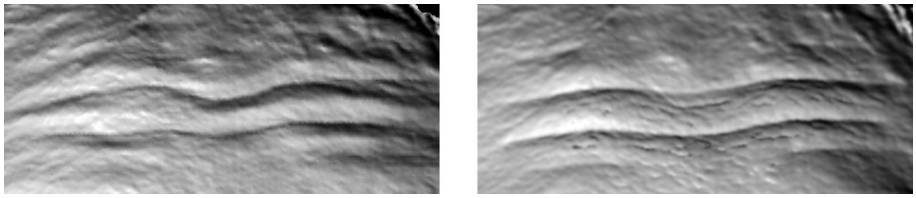


Figure 2.14: A digitized forehead with no wrinkles.



(a) (b)

Figure 2.15: (a) A digitized forehead with wrinkles. (b) Generated wrinkles over the mesh shown in Figure 2.14.

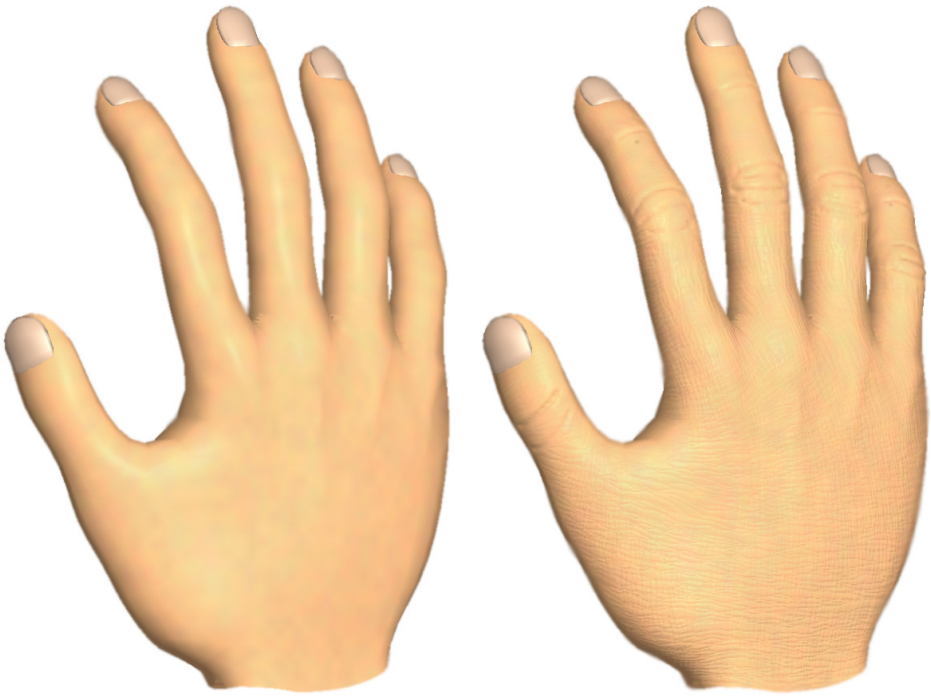
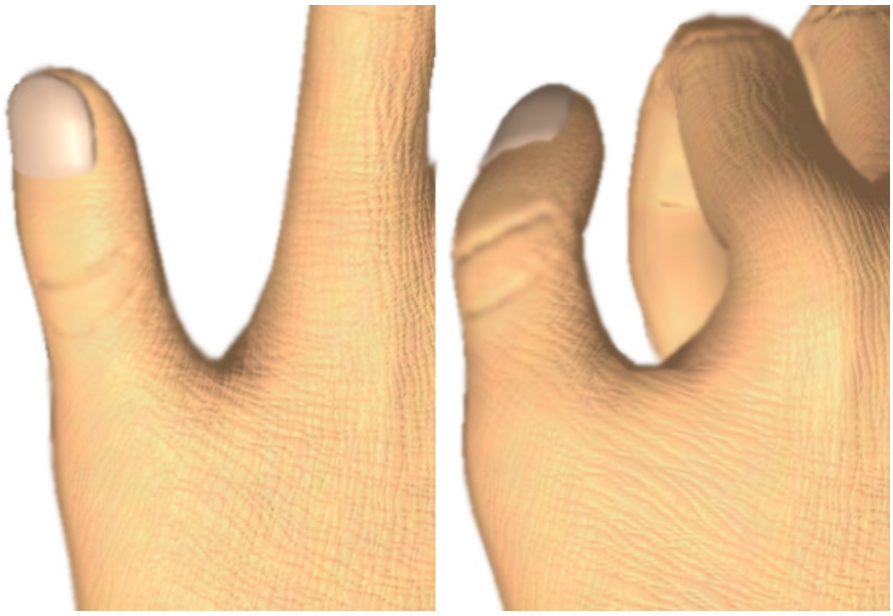


Figure 2.16: The hands with and without wrinkles.



(a) (b)

Figure 2.17: (a) The skin surface of the hand at the rest state. (b) The expanded skin surface.

Figure 2.18: The foot.

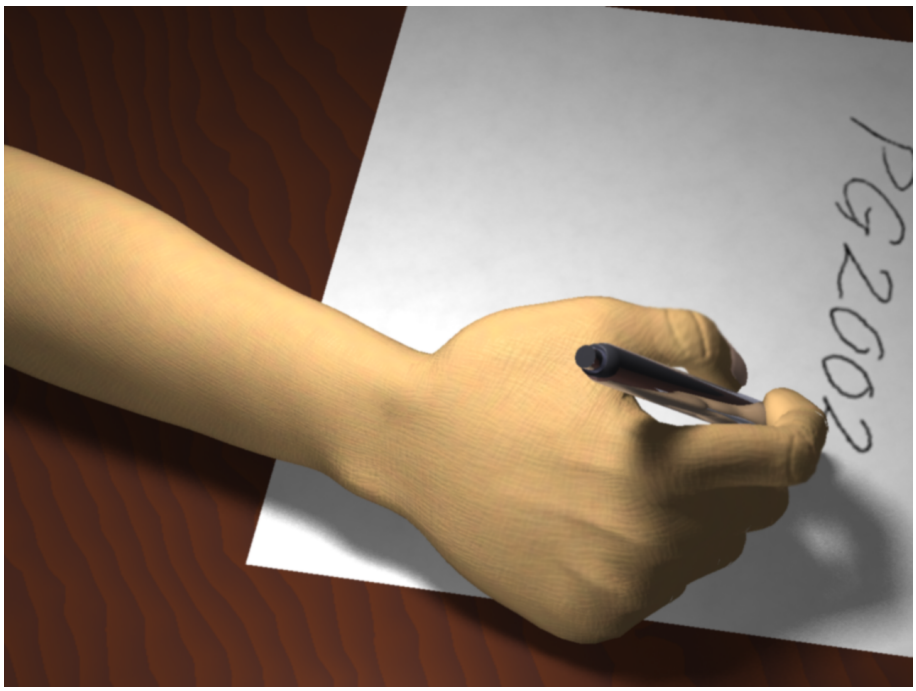


Figure 2.19: The hand with the arm.

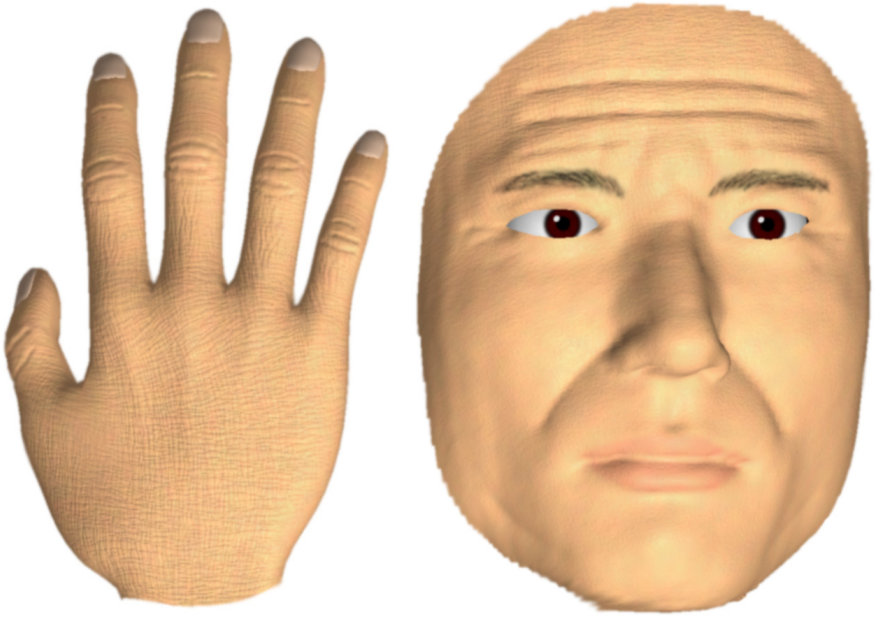


Figure 2.20: The aged skin on the hand and on the face.

- It is easy to implement thanks to its simplicity.
- It performs relatively fast. Wrinkled skin surfaces are rendered at an interactive frame rate.
- It dynamically modulates the amplitude of both fine-scale and large-scale wrinkles according to the skin surface deformation while animating the body part.
- The created wrinkles satisfactorily enhance the realism of skin rendering since our method takes into account the properties of real wrinkles.

We have demonstrated the ability of our method to model realistic wrinkle shapes by comparing them with real wrinkles. This lends support to both the first and the fifth advantages listed above. On the other hand, the limitations of our method are:

- It ignores self-contact of large-scale wrinkles that might occur on the inside of joints. Though this may visually be acceptable because wrinkles are hidden in the folds, it is better to take it into account.
- It provides too much freedom for modeling wrinkles. It is better to let users model wrinkles with some constraints by inferring wrinkle locations and directions from the skin surface deformation.
- It cannot be applied to fine-scale wrinkles on palms and soles because they have different properties from those on the rest of the skin surface.

In addition to addressing the above limitations, the future direciton of this research would be to add more details to further enhance the realism by displaying not only wrinkles, but also other elements on the skin (e.g., pores, hairs, and blood vessels), and also by simulating the subsurface scattering of skin as was done by Jensen et al. [19].

Chapter 3

Modeling of Hair

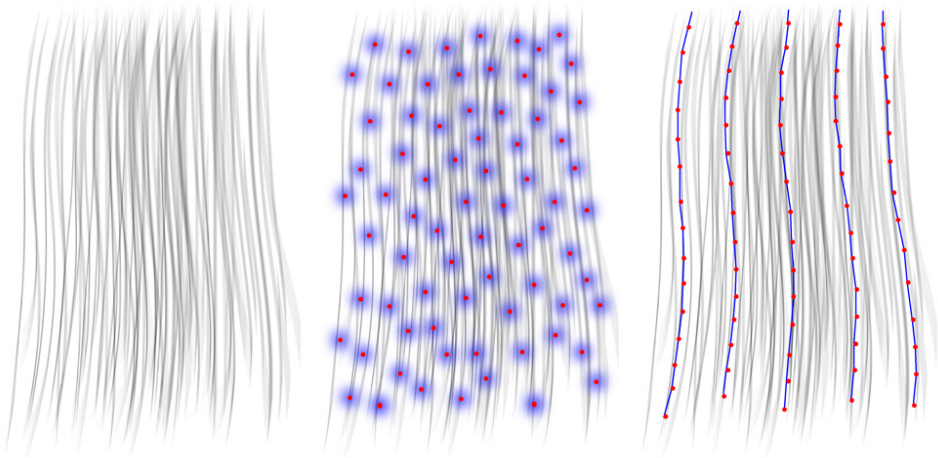
This chapter presents a practical method for modeling hair. We introduce a new model for representing hair called *loosely connected particles (LCP)*, which decreases the high cost of computing the complex interactions among hair.

3.1 Introduction

The ability to represent realistic-looking hair plays a crucial role in the synthesis of life-like human models. However, the enormous number (a human scalp has typically 100,000 strands) and thin nature of hair strands complicate and slow down all of the processes for hair image generation, including modeling, rendering and animation. Moreover, when we want to animate hair based on a physically plausible simulation, the situation is even worse, because we have to take into account interactions among the hair strands (so-called hair-hair interactions) in order to reproduce realistic behavior of the hair, as many researchers have already pointed out [7, 16, 21, 25, 32, 34]. The complex behavior of hair results from the characteristics of individual hair strands and the interactions between them such as collisions, friction, repulsion due to static electricity and cohesion/adhesion due to lipids or hair-dressings [33]. Hair-hair interactions are therefore essential when animating hair. However, computing the interactions among a large number of individual hair strands is still expensive with the computer power currently available.

We propose a practical method for modeling hair that can handle hair-hair interactions at a reasonable cost. The key concept is to model hair as a set of unordered particles that have no tight structure for representing hair strands or clusters, as opposed to the traditional way of modeling hair.

Instead, the individual particles have only loose connections to those nearby (thus we call them *loosely connected particles*, or *LCP*). Note that we neither model the hair strands as trajectories of moving particles nor as serial chains of particles. The particles serve as sampling points of the volume of the hair and the dynamics of the hair, including hair-hair interactions, is simulated using the interacting particles. In our approach, each particle represents a certain amount of the hair medium around it, and it might be viewed as a volume density. This notion is illustrated in Figure 3.1, which was inspired by Hadap and Magnenat-Thalmann [16].



(a) (b) (c)

Figure 3.1: (a) The hair that we wish to represent. (b) The particles are distributed as sampling points of the volume of the hair. (c) Traditional methods represent the hair as serial chains of particles (the same number of particles as (b)).

Representing hair by using sampling points can be regarded as an approximation or simplification of a collection of individual hair strands in order to reduce the high computational cost of checking and evaluating hair-hair interactions. From this point of view, one might think of a more typical method, i.e., to bring several strands together to form a cluster or wisp and to then model each cluster as some structured element, such as a generalized cylinder. This is justifiable because the hair strands tend to clump together, which is one of the frequently observed phenomena resulting from hair-hair interactions, and clustering effects are therefore important for producing natural-looking hair images. However, as Kim and Neumann [22] pointed out, clustering is not a static phenomenon. When hair moves, the hair-hair

interactions cause clusters to split and merge dynamically. To simulate this, we could subdivide the structured elements to construct a hierarchical tree structure of clusters, as in their *multi-resolution hairstyle modeling (MHM)* system. However, it is unclear how we can split and merge the clusters at different levels in different sub-trees. Although we can animate the highest resolution clusters individually, this severely limits the number of clusters available in order to handle hair-hair interactions with a reasonable computation time.

For this reason, we do not arrange the particles to form a fixed set of clusters, but distribute them in an unordered manner throughout the volume of the hair. To simulate dynamic clustering effects, the neighboring particles should have coherence and move together to some extent. Moreover, the neighboring particles should also be able to draw apart, since they do not represent portions of the hair volume that are occupied by exactly the same set of hair strands. Therefore, some looseness in the connections between the particles is essential. We also incorporate a mechanism into these loose connections that maintains the tensile stiffness of the hair by keeping the elastic modulus of each particle constant in order not to lose lengthwise coherence of the hair. Results show that our LCP approach can successfully reduce the computational cost for animating hair without losing most of the characteristics resulting from hair-hair interactions.

The rest of this chapter is organized as follows. Section 3.2 presents a brief review of closely related work. Section 3.3 describes how to model and animate hair with particles, taking into account hair-hair interactions and also making the connections between the particles loose in order not to prevent free lateral motion of the hair. Section 3.4 describes a method for rendering hair that suits our model. Section 3.5 describes a method for handling collisions between the hair and the human body model such as the head and the torso. Section 3.6 shows results and Section 3.7 discusses the advantages and the limitations of our approach.

3.2 Related Work

In this section, we limit our review to any previous work on the animation of hair that takes into account hair-hair interactions. Refer to [36] for a more general and detailed survey.

Several works have modeled some specific aspects of hair-hair interactions. Kim and Neumann [21] added some constraints to hair strands by

enclosing a hair surface within a thin bounding volume. Lee and Ko [25] gave hair body by prohibiting hair strands from falling inside the layers of the head hull.

Hadap and Magnenat-Thalmann [16] animated hair using sufficiently generalized hair-hair interactions. They assumed that hair is a continuum and elegantly handled hair-hair interactions using particle-based fluid dynamics. However, the computational cost was still high because they modeled individual hair strands explicitly with serial chains of rigid segments (10,000 strands with 30 segments each) and glued several particles to each of the segments, so that the amount of computation was huge (2 minutes per frame). The results were excellent, but not to the extent that they captured the discontinuities of hair. Our LCP approach can be viewed as freeing the particles from these serial chains.

Koh and Huang [23] modeled hair clusters as strips of spline surfaces and avoided collisions by introducing springs among them. Plante et al. [32] simulated interactions among the hair clusters, each of which was modeled as a skeleton and its envelope. A skeleton captures the lengthwise shape of the cluster, whereas its envelope captures the cross-sectional shape deformation. Chang et al. [7] simulated only a small number of hair strands called guide hairs, and interpolated the rest. They handled hair-hair interactions with auxiliary triangle strips spanning the nearby strands. These three methods are similar in spirit. They reduce the number of hair strands to be computed (in the form of strips, skeletons, and guide hairs, respectively) in order to speed up the simulation and also to capture the discontinuities of hair. Moreover, the gaps among the sparse hair strands are compensated for by additional structures such as springs, envelopes and auxiliary triangle strips, respectively. We also take a similar approach, but since the particles are distributed over the hair volume as shown in Figure 3.1(b), the gaps between the particles are small. Besides, compensation for these gaps, which actually models hair-hair interactions, is performed through particle dynamics rather than by introducing additional structures. The method by Koh and Huang [23] achieved real-time animation, but the springs accounting for hair-hair interactions were simplistic and imposed strong restrictions on free lateral motion of the hair. The methods proposed by Plante et al. [32] and Chang et al. [7] took up to tens of seconds to obtain the final rendered image.

3.3 Modeling Hair with LCP

Hair is modeled as a set of particles that serve as sampling points of the volume of the hair. Particle i has mass m_i, position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$, just like a standard particle system. The mass indicates the amount of hair medium that the particle represents and the distribution of the particle mass in space determines the density of the hair. The density here indicates the amount of hair medium in a unit volume, as distinguished from the density of the hair material. As described by Hadap and Magnenat-Thalmann [16], we borrow the idea from *smoothed particle hydrodynamics (SPH)* [27, 10], and the density ρ_i at position $\mathbf{x}_i$ is computed as follows:

$$\rho_i = \sum_j m_j W(h, ||\mathbf{x}_j - \mathbf{x}_i||), \tag{3.1}$$

where j runs through all of the particle indices and W represents an interpolating kernel called a *smoothing kernel*. We assume that the particles are smeared out in space so that each particle represents the volume density around it. h is called the *smoothing length* and indicates the extent of the influence. We used the smoothing kernel described in [27] with $W(h, r) = 0$ for $r > 2h$. Thus only nearby particles within a distance of $2h$ contribute to the summation of Equation 3.1.

In addition, particle i has a unit direction vector $\mathbf{t}_i$, which is the direction of hair, and it also has constant coordinates (u_i, v_i, s_i) indicating the initial state of the hair, where (u_i, v_i) are parameters on the scalp and s_i is the arc length of hair from the scalp. The next subsection describes how to determine the initial hair directions and assign these coordinates. The method for updating the hair directions is described in Section 3.3.2.

The dynamics of hair comes from a combination of the characteristics of hair strands and hair-hair interactions. The dynamics of hair strands can be accounted for by forces acting between a fixed set of particle pairs that are initially near neighbors, because two particles in each pair share a portion of some hair strands and should have lengthwise coherence. The method used to determine this set of pairs is also described in the next subsection. On the other hand, forces due to hair-hair interactions act among current neighboring particles. Thus, at each time step we have to search for pairs of particles less than $2h$ apart. This can be performed efficiently by a grid of voxels of size $2h$. Assuming that the distribution of particles is fairly even, the time complexity bears a linear relationship to the number of particles [10].

3.3.1 Particle Initialization

This subsection describes the method for distributing particles over a polygonal head model prior to animation. First, as shown in Figure 3.2(a), we trim the scalp surface from the head model with four Catmull-Rom spline curves [6] by specifying their control points interactively. Then, we embed the scalp surface in a unit square domain $D : [0,1] \times [0,1]$ with axes u and v as shown in Figure 3.2(b), using a piecewise linear approximation of *harmonic mapping* [13].

If we denote this mapping from world space to the parameter domain by $\phi : \Re^3 \rightarrow D$, the scalp surface in the world space is fully parameterized by coordinates (u, v) as $\phi^{-1}(u, v)$. We add a third dimension s over D, which represents the arc length of hair, and then distribute the particles within the volume of this square pillar, as shown in Figure 3.2(c). Its lower bound is the plane $s = -s_r$, where s_r is the depth of the hair root beneath the scalp. The particles with negative arc length are referred to as *root particles*, and are fixed to the head during animation. The upper bound of the parameter volume is given as the hair length map $L(u, v)$. The hair length map is given as a user-defined grayscale image scaled by the maximum length of hair. As we want the particles to evenly sample the volume of the hair, we distribute the particles based on the Poisson disc distribution. In order to compensate for the distortion of mapping ϕ, we use the distance between the corresponding points on the scalp in the world space for the (u, v) coordinates. That is, we define the distance between two points $P_1(u_1, v_1, s_1)$ and $P_2(u_2, v_2, s_2)$ in the parameter domain as:

$$d(P_1, P_2) = \sqrt{||\phi^{-1}(u_1, v_1) - \phi^{-1}(u_2, v_2)||^2 + (s_1 - s_2)^2}. \tag{3.2}$$

Finally, particle i with parameters (u_i, v_i, s_i) is mapped back to the world space $\mathbf{x}_i$ using the direction vector of the hair root $\mathbf{t}_r(u, v)$ defined over the scalp as:

$$\mathbf{x}_i = \phi^{-1}(u_i, v_i) + s_i \frac{\mathbf{t}_r(u_i, v_i)}{||\mathbf{t}_r(u_i, v_i)||}. \tag{3.3}$$

Specifying the root direction $\mathbf{t}_r(u, v)$ manually is rather a tedious task, and we can determine the root direction simply by the following steps. The user first selects a point on the scalp as a whirl of hair. We denote the normal vector at this point by $\mathbf{n}_w$. Then the root direction is computed using the unit normal vector $\mathbf{n}(u, v)$ of the scalp by the following equation:

$$\mathbf{t}_r(u, v) = -w_t\{\mathbf{n}_w - (\mathbf{n}_w \cdot \mathbf{n}(u, v))\mathbf{n}(u, v)\} + w_n\mathbf{n}(u, v), \tag{3.4}$$

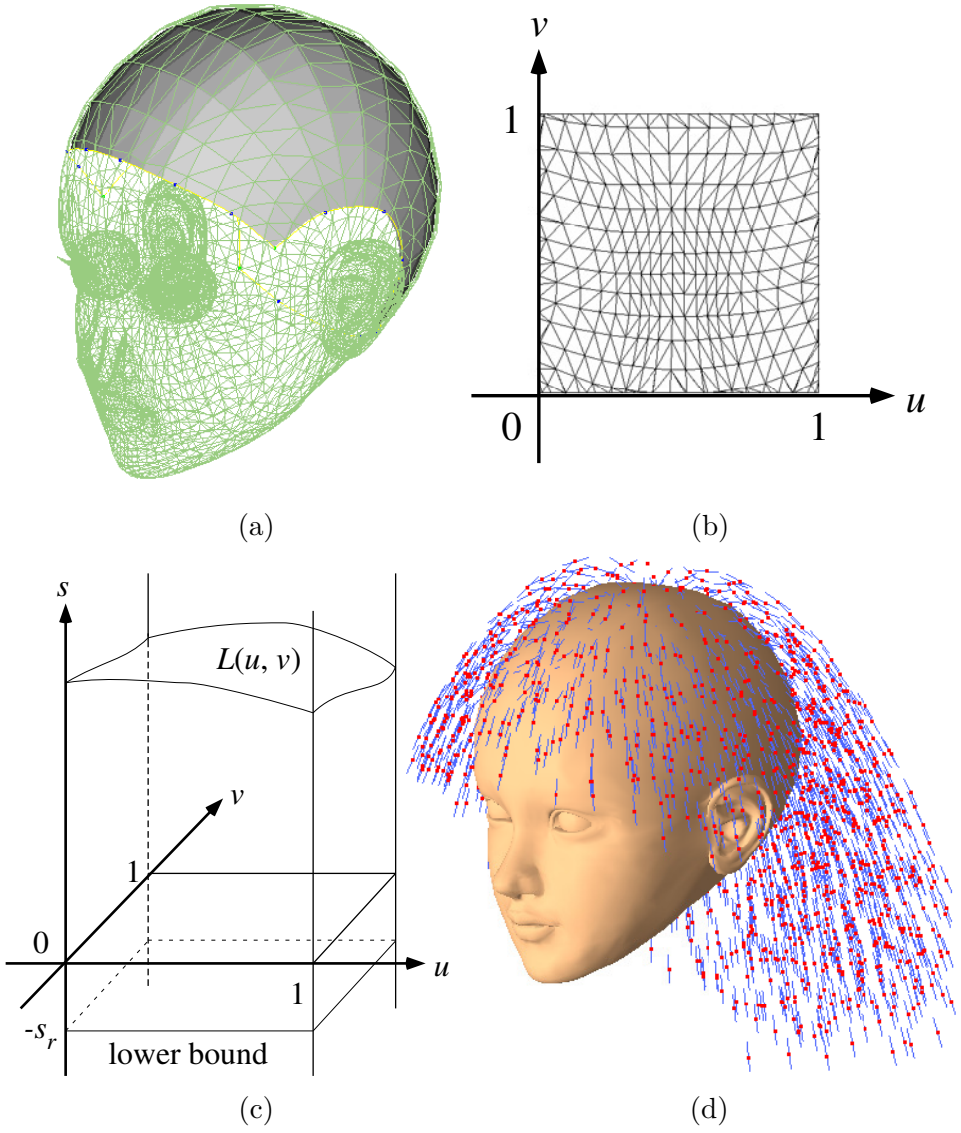


Figure 3.2: Initializing particles over the scalp. (a) A trimmed scalp surface. (b) The scalp surface embedded in a unit square. (c) The volume of the parameter domain (u, v, s). (d) The particles mapped back to the world space. The line segments indicate the hair directions.

where w_t and w_n are weighting constants for the tangent and normal components of the hair root direction, respectively.

To make the initial state of the hair more natural, we apply the cantilever beam simulation method proposed by Anjyo et al. [1]. Though our hair model does not rely on serial chains of rigid segments, we can treat particle i with parameters (u_i, v_i, s_i) except for the root particles as the endpoint of such a chain of length s_i emanating from its root position $\phi^{-1}(u_i, v_i)$ on the scalp. The direction $\mathbf{t}_i$ of the particle is matched to the last segment of the chain. Figure 3.2(d) shows the resultant initial state of the particles.

To simulate the dynamics of hair strands, we establish the connections between nearby particles. As shown in Figure 3.3(a), the neighbor search is performed with a smoothing length h_s for the positions of the particles initialized in the above procedure, where h_s is a little larger than the initial average inter-particle distance. Moreover, we store the intial state of each particle pair (i, j) by computing the distance l_{ij}^0 between the two particles, and the angle θ_{ij}^0 between the hair direction $\mathbf{t}_i$ and the direction of the particle pair $\mathbf{x}_{ij}$ as shown in Figure 3.3(b), where $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$.

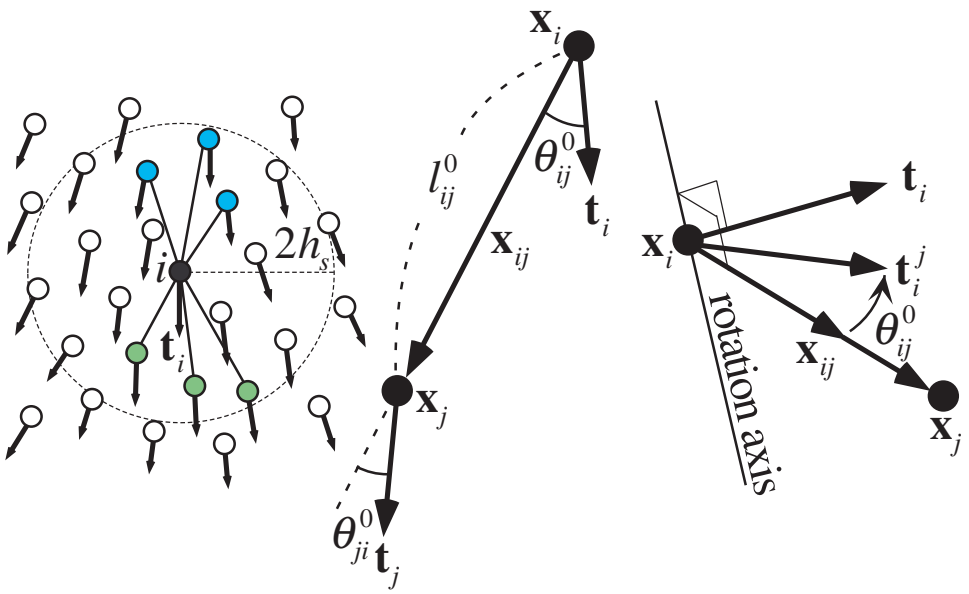
We assign a strength of connection c_{ij} to the particle pair using the following equation, which is used for the spring coefficients described in the next subsection:

$$c_{ij} = a_{ij} l_{ij}^0, \tag{3.5}$$

where a_{ij} indicates the degree to which two particles are aligned in a row and thus how strong their lengthwise coherence is. This is computed as follows:

$$a_{ij} = \begin{cases} |(\cos\theta_{ij}^0 + \cos\theta_{ji}^0)/2| & \cos\theta_{ij}^0 \cdot \cos\theta_{ji}^0 > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3.6}$$

Because we activate the spring forces that account for the stiffness of the hair strands, connecting particles in close proximity with stiff springs makes the system unstable. Therefore, the second factor l_{ij}^0 in Equation 3.5 is used for stability. For computational efficiency, we first omit the pairs which have a_{ij} smaller than a threshold a_0. Then, from the pairs which have smaller values for $c_{ij} W(h_s, l_{ij}^0)$, which is the initial spring coefficient between the two particles as will be described in Equation 3.7 in the next subsection, we omit them recursively until the number of connections for each particle is around the specified value N_n. Experimentally we use $a_0 = 0.5$ and $N_n = 12$. In Figure 3.3(a), $N_n = 6$.



(a) (b) (c)

Figure 3.3: (a) The neighbor search is performed with a smoothing length h_s (so the radius is set as $2h_s$), and the particles connected with the center particle i are divided into two groups: $N_r(i)$ (the blue nodes) and $N_t(i)$ (the green nodes). (b) The initial state of a particle pair. (c) A diagram for updating the hair direction.

3.3.2 Dynamics of Hair

In this subsection, we focus on the dynamics of hair that is due to the characteristics of hair strands, and hair-hair interactions are explained in the next subsection.

The hair volume can be considered as a deformable body, which can be animated by connecting the neighboring particles with damped springs. The particle pairs are established in the initialization process described in the previous subsection and this set of pairs is fixed during the animation phase. To realize free lateral motion of the hair, we make the connections between the particles loose by using the following nonlinear spring coefficient:

$$k_{ij} = c_{ij} W(h_s, ||\mathbf{x}_{ij}||), \tag{3.7}$$

which gradually decreases as the two particles draw apart until they are completely free from each other at a distance of $2h_s$.

However, these loose connections also allow the hair to stretch. Therefore, in order to prevent the tensile stiffness of hair from changing, we keep the tensile elastic modulus of each particle constant during animation. The elastic modulus E is unique to a material, and we can relate it to the rest length L and a cross-sectional area A of the material by the following equaiton:

$$E = \frac{L}{A} K, \tag{3.8}$$

where K corresponds to the spring coefficient when we view the material as the lengthwise spring. In the following, we express E in terms of the parameters of the particles and the springs among them. The cross-sectional area A_i of the hair which paricle i represents (not of individual hair strands) can be written as the following equation using the denstiy δ of the hair material:

$$A_i = \left(\frac{m_i}{\delta} \right)^{\frac{2}{3}}. \tag{3.9}$$

Since we are concerned with the tensile stiffness of the hair, we divide the neighboring particles of particle i into two groups, as shown in Figure 3.3(a): one includes those closer to the root of the hair, and the other includes those closer to the tip of the hair. They are denoted by $N_r(i)$ and $N_t(i)$, respectively. For each group, we sum up all of the contributions of the springs to the elastic modulus of particle i as follows:

$$E_{r,i} = \frac{1}{A_i} \sum_{j \in N_r(i)} l_{ij}^0 \alpha_{ij} k_{ij} \tag{3.10}$$

$$E_{t,i} = \frac{1}{A_i} \sum_{j \in N_t(i)} l^0_{ij} \alpha_{ij} k_{ij}, \tag{3.11}$$

where l^0_{ij} is the rest length of the spring between particles i and j, which is their initial distance as described in the previous subsection, and α_{ij} is an unknown scaling factor for the spring coefficient that we have to solve to keep $E_{r,i}$ and $E_{t,i}$ equal to the elastic modulus of the hair. The number of unknowns α_{ij} is the number of particle pairs, and it is much larger than the number of equations, which is twice the number of the paricles. Thus, Equations 3.10 and 3.11 form an underdetermined sparse linear system. To keep the computational cost as low as possible, we directly (non-iteratively) obtain an approximate solution to this system in linear time in the number of particles, which cannot be accomplished by using precise direct methods for underdetermined linear systems [8]. By letting

$$\alpha_{ij} = \alpha_{r,i} \quad \text{for} \quad \forall j \in N_r(i), \tag{3.12}$$

we solve Equation 3.10 for each i individually as:

$$\alpha_{r,i} = \frac{E_{r,i} A_i}{\sum_{j \in N_r(i)} l^0_{ij} k_{ij}}. \tag{3.13}$$

Similarly Equation 3.11 is solved to obtain $\alpha_{t,i}$, and we take the average of the two solutions as:

$$\alpha_{ij} = \begin{cases} (\alpha_{r,i} + \alpha_{t,j})/2 & i \in N_t(j) \wedge j \in N_r(i) \\ (\alpha_{r,j} + \alpha_{t,i})/2 & i \in N_r(j) \wedge j \in N_t(i) \end{cases}. \tag{3.14}$$

Though this is a rough approximation, the tensile stiffness and coherence were visually well maintained in our experiments.

So far we have mentioned the tensile stiffness of the hair. The bending stiffness of the hair is also accounted for by the same springs since we model the hair volume as an anisotropically deformable body. Therefore, both the tensile and the bending stiffness will increase when we increase either the elastic modulus of hair or the particle mass. However, there are two factors that affect the particle mass, the cross-sectional area and the number of hair strands. And in the latter case, even if we increase the number of hair strands, the bending stiffness does not increase as much because they can slide over each other. Thus we adjust the bending stiffness relative to the tensile stiffness by scaling the repelling force when the springs are compressed. We do not consider the twist of hair, and neglect the torsional stiffness.

Based on the current spring coefficient $\alpha_{ij}k_{ij}$ of the connection between the particles, we update the hair direction $\mathbf{t}_i$ of particle i. As described in the previous subsection, particle i has the initial angle θ_{ij}^0 between the hair direction and the direction of the connection. Thus, we can compute the updated hair direction $\mathbf{t}_i^j$ according to the current hair direction $\mathbf{t}_i$ and the direction of the connection $\mathbf{x}_{ij}$ by rotating the vector $\mathbf{x}_{ij}$ with angle θ_{ij}^0 around the axis normal to both $\mathbf{t}_i$ and $\mathbf{x}_{ij}$ as shown in Figure 3.3(c). Taking the weighted average, the final updated hair direction is:

$$\mathbf{t}_i^* = \sum_j \alpha_{ij} k_{ij} \mathbf{t}_i^j, \tag{3.15}$$

where $\mathbf{t}_i^*$ is normalized subsequently.

3.3.3 Hair-hair Interactions

We model attraction/repulsion, collision and friction to account for hair-hair interactions. The forces due to these interactions act upon current nearby particles, and can be modeled by using a smoothing length h that is a little smaller than the initial average inter-particle distance.

Attraction/repulsion forces can be caused by lipids, hair-dressings or static electricity. Macroscopically, the effect of these interactions is to preserve the average density of hair. The force on particle i due to particle j is given by the following equation, which is the pressure force defined in SPH [27].

$$\mathbf{f}_{a,ji} = -m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \frac{\partial}{\partial \mathbf{x}_i} W(h, ||\mathbf{x}_{ij}||), \tag{3.16}$$

where the pressure P is modeled as $P = k_a(\rho - \rho_0)$ and k_a and ρ_0 control the magnitude of the force and the average density, respectively [10]. A small value of ρ_0 makes the hair repulsive, whereas a large value makes it cohesive.

To model inelastic collisions of hair, we simply reduce the relative velocity $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$ of two particles if they are approaching. The direction of the collision, $\mathbf{d}_n$, is normal to the hair directions $\mathbf{t}_i$ and $\mathbf{t}_j$ of both particles. Thus, $\mathbf{d}_n = \mathbf{t}_i \times \mathbf{t}_j$. If $||\mathbf{d}_n|| \ll 1$, the two hair directions are almost colinear, and in this case we can compute the direction of the collision as $\mathbf{d}_n = \mathbf{t}_i - (\mathbf{t}_i \cdot \hat{\mathbf{x}}_{ij})\hat{\mathbf{x}}_{ij}$, where the notation $\hat{\mathbf{x}}$ denotes a unit vector in the same direction as $\mathbf{x}$. The force is then computed as follows:

$$\mathbf{f}_{c,ji} = \begin{cases} d_c \; W(h, ||\mathbf{x}_{ij}||) \; (\mathbf{v}_{ij} \cdot \hat{\mathbf{d}}_n)\hat{\mathbf{d}}_n & (\mathbf{v}_{ij} \cdot \mathbf{d}_n)(\mathbf{x}_{ij} \cdot \mathbf{d}_n) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{3.17}$$

where d_c is the collision damping constant.

Friction is modeled similarly. The direction $\mathbf{d}_t$ of the frictional force is defined as $\mathbf{v}_{ij} - (\mathbf{v}_{ij} \cdot \mathbf{d}_n)\mathbf{d}_n$, which is normal to the direction of the collision $\mathbf{d}_n$, and the force is computed as:

$$\mathbf{f}_{f,ji} = d_f \; W(h, ||\mathbf{x}_{ij}||) \; (\mathbf{v}_{ij} \cdot \hat{\mathbf{d}}_t)\hat{\mathbf{d}}_t, \tag{3.18}$$

where d_f is the frictional damping constant.

3.4 Rendering Hair

Since the particles represent the volume density of hair, we can render hair by using volume rendering techniques, just as Kajiya and Kay did for their furry teddy bear [20]. Unlike the standard voxel grid structure, the density field in our model is represented by the distribution of particles, and in this case we can benefit greatly from the splatting method with billboards by using the hardware accelerated method presented by Dobashi et al. [11]. We assume a parallel light source, and in the first pass, we project the particles in the direction of the light source in order to ascertain how much of the light intensity is attenuated. In the second pass, we can then render the hair with complex self-shadows that enhance the volumetric appearance of the hair.

The following is the pseudo-code for rendering hair and other objects, such as the head and the torso, which includes computation of shadows cast on the hair by the objects and by the hair itself, and those cast on the objects by the hair and by the objects themselves.

```
// compute shadow
1. Clear the color buffer in white and clear the depth buffer.
2. Transform the scene to be viewed from the light source.
3. Enable depth-buffering.
4. Render the objects in white.
5. Read the depth buffer for shadow mapping.
6. Disable depth-buffering.
7. Sort the particles in ascending order of depth.
8. for each particle not in the objects' shadows
9.     Read the intensity of the pixel where the particle is projected.
10.    Render the particle with a billboard.
11. end for
```

12. Read the color buffer for light mapping.

// render scene
13. Clear the color buffer and the depth buffer.
14. Transform the scene to be viewed from the viewpoint.
15. Enable depth-buffering.
16. Render the objects with shadow and light mapping.
17. Write-disable the depth buffer.
18. Sort the particles in descending order of depth.
19. **for each** particle
20. Render the particle with a billboard using the intensity of the pixel.
21. **end for**

The shadow map obtained in Line 5 is used for computing shadows cast by the objects in Line 16. The intensity read from the color buffer in Line 9 represents the intensity of the light reaching the particle attenuated through the hair. This value is used in Line 20 for self-shadows of the hair. The light map obtained in Line 12 represents the intensity of the light penetrates the hair that is not in the shadows cast by the objects. This is used for shadows cast on the objects by the hair in Line 16. Figure 3.4 shows the rendered images of our hair model without and with shadows.

(a) (b)

Figure 3.4: The hair rendered (a) without and (b) with shadows.

To simulate the highly anisotropic phase function of hair we use the Goldman model [14], which adds a directionality factor to the Kajiya model

[20] that controls the reflection and transmission ratios of light. However, as the number of particles is relatively small for computational efficiency, the phase function alone is not sufficient to produce the appearance of hair strands. We compensate this loss by mapping hair texture to the billboards instead of the simple texture of a Gaussian distribution. Figure 3.5(a) shows an example of hair texture. We match the hair direction of the texture (usually the vertical axis) to the particle direction $\mathbf{t}_i$ as shown in Figure 3.5(b). As opposed to the standard billboard, we can no longer place it in parallel with the screen. Hence, we scale the opacity of the billboard by $1/\sin\theta$, where $\cos\theta = \mathbf{t}_i \cdot \mathbf{d}_p$ and $\mathbf{d}_p$ is the direction of projection. We can further augment the directionality by scaling the size of the billboards as shown in Figure 3.5(c).

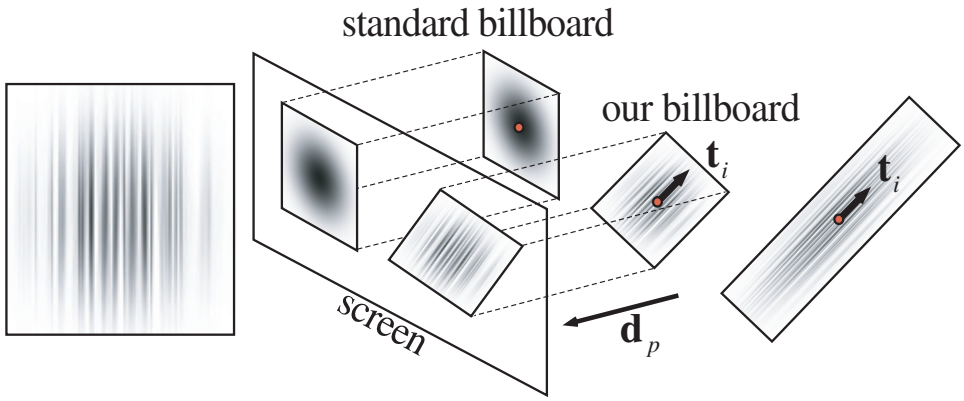


Figure 3.5: (a) An example of hair texture. (b) The hair direction of the texture is matched to the particle direction. This figure shows the case of orthogonal projection. The case of perspective projection is similar. (c) The size of the billboard is scaled.

3.5 Hair-body Collision

Collisions between the particles and the human body model are detected efficiently by using the voxel grid structure. For each voxel we compute a plane that locally approximates the shape of the body. We first sample points randomly over the surface of the body model. For uniformity, the

number of the sampling points on each face is proportional to its area. Then we average their positions and normals in each voxel, and obtain position $\mathbf{p}$ and normal $\mathbf{n}$ that define the local plane of the body, as shown in Figure 3.6. The voxels that contain no sampling points are considered distant from the surface of the body, and therefore we do not check collisions within them. These planes are computed only once in the initialization phase and fixed during animation as long as the body is in rigid motion (composed of only rotations and translations).

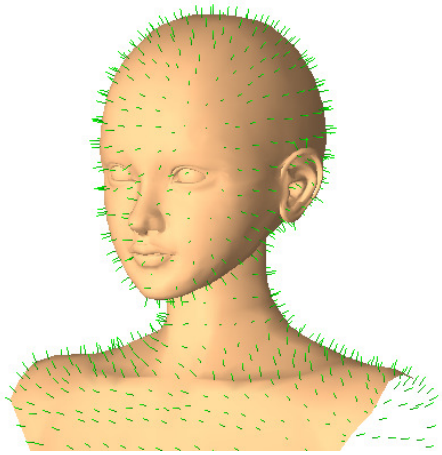


Figure 3.6: Green line segments show the normals of the surface of the body emanating from the average position of the sampling points within each voxel.

Whether a particle at position $\mathbf{x}$ is in contact with the body is determined by evaluating its distance d from the surface of the body.

$$d = (\mathbf{x} - \mathbf{p}) \cdot \mathbf{n}. \tag{3.19}$$

If d is smaller than a certain threshold, which is around the smoothing length h, we perform the following collision handling procedure. We first decompose the particle's relative velocity $\mathbf{v}$ to the body and the total applied force $\mathbf{f}$ into the normal and tangential components. These are denoted by $\mathbf{v}_n$, $\mathbf{v}_t$, $\mathbf{f}_n$ and $\mathbf{f}_t$, respectively. Then we compute the normal velocity after time step Δt.

$$\mathbf{v}'_n = \mathbf{v}_n + \mathbf{f}_n \Delta t. \tag{3.20}$$

If $\mathbf{v}'_n \cdot \mathbf{n} > 0$, the particle is going away from the body, so we leave it as it is. Otherwise we apply force $\mathbf{f}_c$ to stop the particle at the surface of the body

in order to model the inelastic collision as follows:

$$\mathbf{f}_c = -\mathbf{v}'_n/\Delta t = -\mathbf{f}_n - \mathbf{v}_n/\Delta t. \tag{3.21}$$

Moreover, we apply frictional force to the particle according to Coulomb's model with a single friction parameter μ as in [5]. The tangential velocity after time step Δt without friction is:

$$\mathbf{v}'_t = \mathbf{v}_t + \mathbf{f}_t \Delta t. \tag{3.22}$$

The frictional force is exerted in the opposite direction of $\mathbf{v}'_t$ with a magnitude at most $\mu||\mathbf{f}_c||$, noting that the particle and the body are pressing each other with the force $\mathbf{f}_c$. If $||\mathbf{v}'_t|| < \mu||\mathbf{f}_c||\Delta t$, the particle will stop slipping or remain stuck. Thus, the frictional force $\mathbf{f}_f$ is computed as follows:

$$\mathbf{f}_f = \begin{cases} -\mathbf{v}'_t/\Delta t & ||\mathbf{v}'_t|| < \mu||\mathbf{f}_c||\Delta t \\ -\mu||\mathbf{f}_c||\mathbf{v}'_t/||\mathbf{v}'_t|| & \text{otherwise} \end{cases}. \tag{3.23}$$

Finally, since there is no guarantee that the particle will not penetrate the surface of the body, we apply repulsion force $\mathbf{f}_r$ to the particle at a negative distance d from the surface as follows:

$$\mathbf{f}_r = -k_r d\mathbf{n}, \tag{3.24}$$

where k_r controls the magnitude of the force.

3.6 Results

We successfully animated the hair with the head and torso model at an interactive frame rate. In addition to the forces described in Section 3.3.2 and Section 3.3.3, we apply gravity, air friction and inertia force due to head movement to each particle. We integrated the equations of motion of the particles with a leapfrog integrator.

Figures 3.7(a) $\sim$ (c) show a few frames from an animation of the back hair. The hair is forced to split into two, and then merge again because of gravity. Hair-hair interactions prevent the hair from penetrating each other as shown in Figure 3.7(c), which is a contrast to Figure 3.7(d) where hair-hair interactions are not considered.

Figure 3.8 shows an animation of the hair when the head is shaken. The head turns around and the hair follows this motion with delay because of inertia. Dynamic clustering effects are visible.

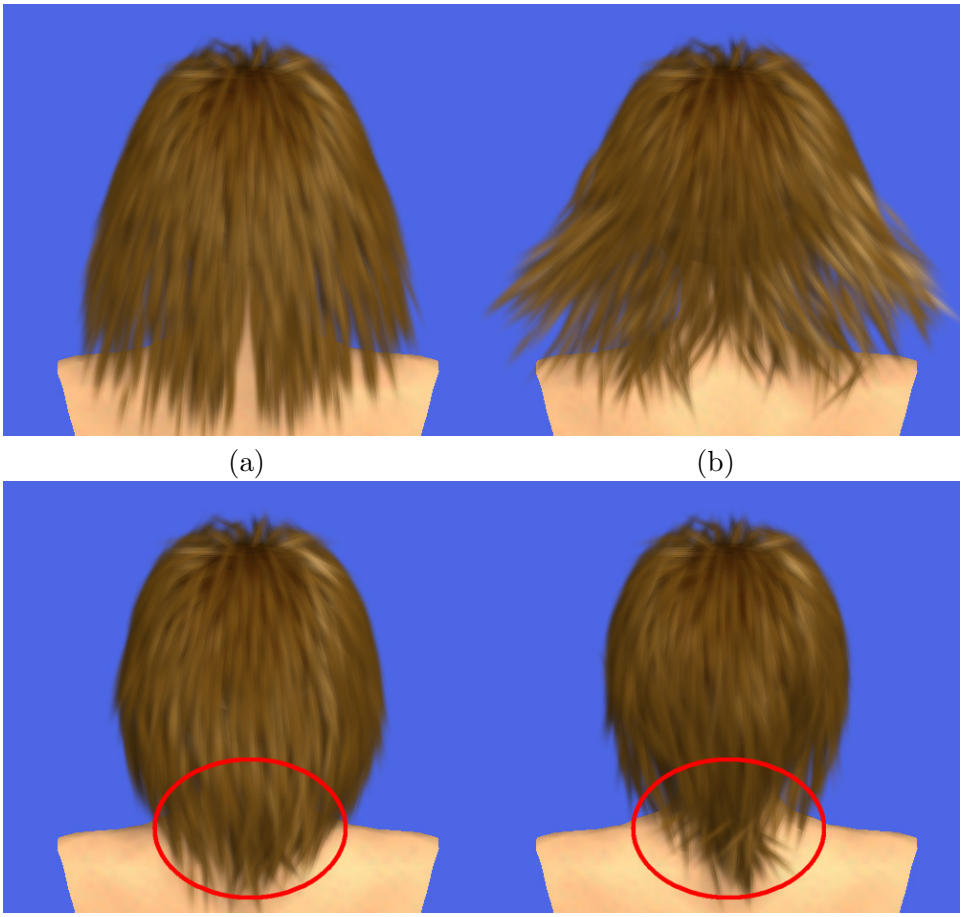


Figure 3.7: (a) ∼ (c) A few frames from an animation of the back hair. (d) The hair penetrates each other when hair-hair interactions are not considered, which is a contrast to (c). The difference is clearly seen in the red circles.

Figure 3.8: Animation of the hair when the head is shaken. Left to right, top to bottom.

Figure 3.9 shows an animation of the hair blowing in wind. The wind is also represented by a set of particles, and is simulated using SPH [27]. The interactions between the hair particles and the air particles are modeled in the same way as Hadap and Magnenat-Thalmann [16] did, and thus the wind is also affected by movement of the hair. Since our LCP approach models hair solely with particles, other particle-based methods are easily incorporated with a slight modification to the programming code.

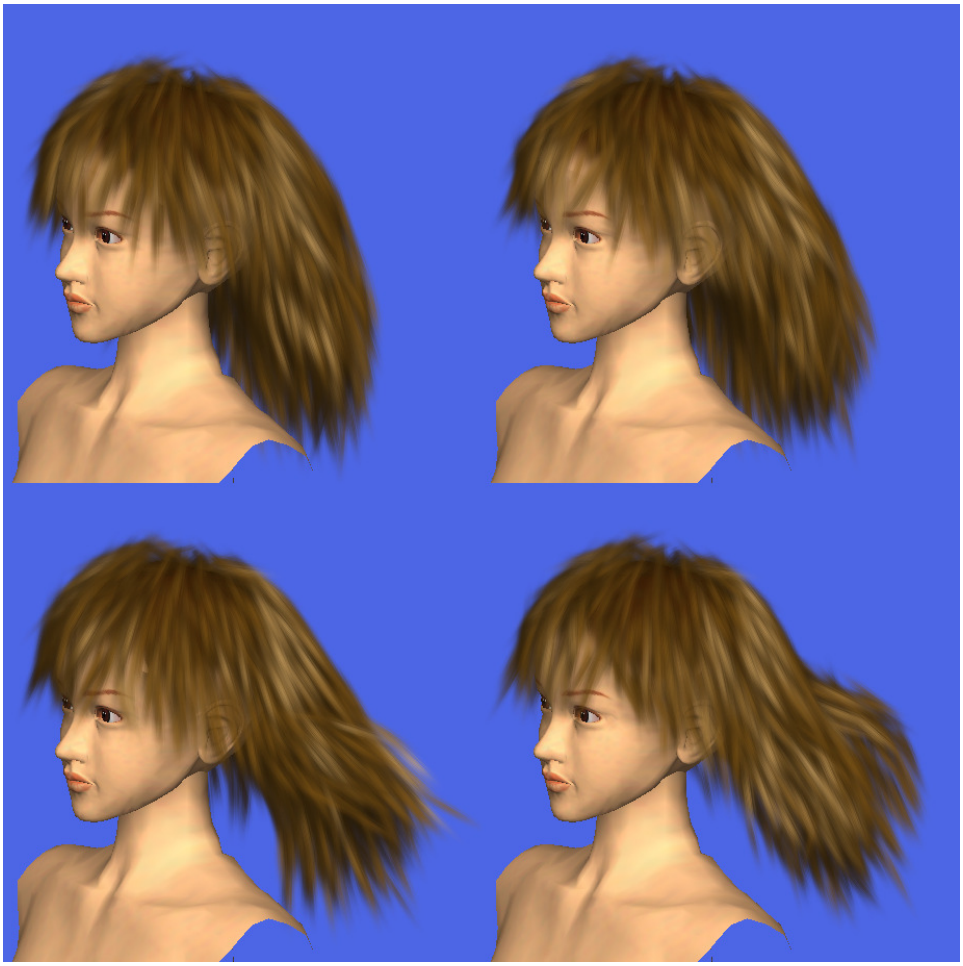


Figure 3.9: Animation of the hair blowing in wind. Left to right, top to bottom.

We used about 2,000 particles to represent the hair for all of the animations presented above. The number of particles is determined by a compromise between accuracy and efficiency, and we chose the value in order to make the animation, including both simulation and rendering with shadows,

can be performed at an interactive frame rate. The hair was animated at 6.7 fps on a PC with an Intel Pentium 4 2.8 GHz CPU and an NVIDIA Quadro4 900 XGL GPU. The simulation took 33% of the total computation time, and the rendering took the rest. The image size and the size of the buffer for computing shadows are 512×512.

3.7 Discussion

In this chapter we have presented a practical model for representing hair called *loosely connected particles (LCP)*. The hair is modeled as a set of unordered particles that serve as sampling points of the volume of the hair. The loose connections between the particles make the hair free to move laterally, and the mechanism for maintaining the tensile stiffness provides lengthwise coherence. Our LCP approach has the following advantages:

- It reduces the high computational cost of simulating the dynamics of the hair by using a fairly small number of particles (about 2,000). Reducing the number of particles does not strongly restrict free lateral motion of the hair since our LCP model does not utilize tight structures for representing hair strands or clusters.
- The dynamics of the hair, including hair-hair interactions, is simulated solely with the interacting particles. No additional structures are needed to account for hair-hair interactions, which simplifies the implementation.

On the other hand, there are limitations to our method:

- The simulation is not physically rigorous. The nonlinear spring coefficients accounting for the stiffness of the hair are designed to obtain the desirable motion of the hair.
- Currently it handles only straight hairstyles. The effectiveness of our method will be greatly improved by making various hairstyles available, such as braids, ponytails and curly hair. It would be nice to incorporate the existing hairstyle modeling techniques into our system.

In addition to addressing the above limitations, the future direction of this work would be to take further advantage of the particle representation, for example, adaptive sampling of the volume of the hair and simulation level of detail [31].

Chapter 4

Conclusion

This thesis has addressed two problems concerning display of virtual humans. Chapter 2 has presented a simple method for modeling wrinkles on human skin and Chapter 3 has presented a practical method for modeling hair. With these methods, we can provide virtual humans with expressive wrinkles and dynamic hair at a reasonable cost, and can make them more realistic and alive. However, using these methods altogether might induce some new problems, such as handling interactions between hair and wrinkles, in addition to the increase in the amount of computation. So far we have been concerned with each aspect of modeling humans individually, and making the whole human model still seems to be a challenging problem. Our methods can reduce the complication of capturing complex and subtle nature of humans, but only a fraction of it. Synthesis of human models remains one of the most difficult things in the field of computer graphics. Therefore, in addition to addressing the future work described in Chapters 2 and 3, we must continue to seek the way to further approach this hard goal.

Bibliography

[1] K. Anjyo, Y. Usami, and T. Kurihara. A Simple Method for Extracting the Natural Beauty of Hair. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26, 2, 111-120, 1992.

[2] Y. Bando, T. Kuratate, and T. Nishita. A Simple Method for Modeling Wrinkles on Human Skin. In *Proceedings of Pacific Graphics 2002*, 166-175, 2002.

[3] J. Blinn. Simulation of Wrinkled Surface. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, 12, 3, 286-292, 1978.

[4] L. Boissieux, G. Kiss, N. M. Thalmann, and P. Kalra. Simulation of Skin Aging and Wrinkles with Cosmetics Insight. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation 2000*, 15-27, 2000.

[5] R. Bridson, R. Fedkiw, and J. Anderson. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Transactions on Graphics*, 21, 3, 594-603, 2002.

[6] E. Catmull and R. Rom. A Class of Local Interpolationg Splines. *Computer Aided Geometric Design (Proceedings of International Conference on Computer Aided Geometric Design '74)*, 317-326, 1974.

[7] J. T. Chang, J. Jin, and Y. Yu. A Practical Model for Hair Mutual Interactions. In *Proceedings of 2002 ACM SIGGRAPH Symposium on Computer Animation*, 73-80, 2002.

[8] R. E. Cline and R. J. Plemmons. l_2-Solutions to Underdetermined Linear Systems. *SIAM Review*, 18, 1, 92-106, 1976.

[9] S. Cotton, E. Claridge, and P. Hall. A Skin Imaging Method Based on a Color Formation Model and its Application to the Diagnosis of Pig-

mented Skin Lesions. In *Proceedings of Medical Image Understanding and Analysis '99*, 49-52, 1999.

[10] M. Desbrun and M. P. Gascuel. Smoothed Particles: A New Paradigm for Animating Highly Deformable Bodies. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation '96*, 61-76, 1996.

[11] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita. A Simple, Efficient Method for Realistic Animation of Clouds. In *Proceedings of SIGGRAPH 2000*, 19-28, 2000.

[12] J. Doyle and J. Philips. *Manual on Experimental Stress Analysis (fifth edition).* Society for Experimental Mechanics, 1989.

[13] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In *Proceedings of SIGGRAPH 95*, 173-182, 1995.

[14] D. Goldman. Fake Fur Rendering. In *Proceedings of SIGGRAPH 97*, 127-134, 1997.

[15] S. Hadap, E. Bangerter, P. Volino, and N. M. Thalmann. Animating Wrinkles on Clothes. In *Proceedings of IEEE Visualization '99*, 175-182, 1999.

[16] S. Hadap and N. M. Thalmann. Modeling Dynamic Hair as a Continuum. In *Computer Graphics Forum (Proceedings of Eurographics 2001)*, 20, 3, 329-338, 2001.

[17] A. Haro, B. Guenter, and I. Essa. Real-time, Photo-realistic, Physically Based Rendering of Fine Scale Human Skin Structure. In *Proceedings of Eurographics Workshop on Rendering 2001*, 53-62, 2001.

[18] T. Ishii, T. Yasuda, S. Yokoi, and J. Toriwaki. A Generation Model for Human Skin Texture. In *Proceedings of Computer Graphics International '93*, 139-150, 1993.

[19] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A Practical Model for Subsurface Light Transport. In *Proceedings of SIGGRAPH 2001*, 511-518, 2001.

[20] J. Kajiya and T. Kay. Rendering Fur with Three Dimensional Textures. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23, 4, 271-280, 1989.

[21] T. Y. Kim and U. Neumann. A Thin Shell Volume for Modeling Human Hair. In *Proceedings of Computer Animation 2000*, 121-128, 2000.

[22] T. Y. Kim and U. Neumann. Interactive Multiresolution Hair Modeling and Editing. *ACM Transactions on Graphics*, 21, 3, 620-629, 2002.

[23] C. K. Koh and Z. Huang. A Simple Physics Model to Animate Human Hair Modeled in 2D Strips in Real Time. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation 2001*, 127-138, 2001.

[24] E. P. F. Lafortune, S. C. Foo, K. E. Torrance, and D. P. Greenberg. Non-Linear Approximation of Reflectance Functions. In *Proceedings of SIGGRAPH 97*, 117-126, 1997.

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

[26] S. R. Marschner, S. H. Westin, E. P. F. Lafortune, K. E. Torrance, and D. P. Greenberg. Image-Based BRDF Measurement Including Human Skin. In *Proceedings of Eurographics Workshop on Rendering '99*, 139-152, 1999.

[27] J. J. Monaghan. Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30, 543-574, 1992.

[28] W. Montagna, A. M. Kligman, and K. S. Carlisle. *Atlas of Normal Human Skin.* Springer, 1992.

[29] S. L. Moschella and H. J. Hurley. *Dermatology (third edition)*, volume 1. W. B. Saunders Company, 1992.

[30] M. Nahas, H. Huitric, M. Rioux, and J. Domey. Facial Image Synthesis Using Skin Texture Recording. *The Visual Computer*, 6, 6, 337-343, 1990.

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

[32] E. Plante, M. P. Cani, and P. Poulin. A Layered Wisp Model for Simulating Interactions Inside Long Hair. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation 2001*, 139-148, 2001.

[33] R. Robbins. *Chemical and Physical Behavior of Human Hair (second edition)*. Springer-Verlag, 1988.

[34] R. Rosenblum, W. Carlson, and E. Tripp. Simulating the Structure and Dynamics of Human Hair: Modeling, Rendering and Animation. *The Journal of Visualization and Computer Animation*, 2, 4, 141-148, 1991.

[35] D. Terzopoulos and K. Waters. Physically-Based Facial Modeling, Analysis, and Animation. *Journal of Visualization and Computer Animation*, 1, 4, 73-80, 1990.

[36] N. M. Thalmann, S. Hadap, and P. Kalra. State of the Art in Hair Simulation. In *Proceedings of International Workshop on Human Modeling and Animation*, 3-9, 2000.

[37] M. L. Viaud and H. Yahia. Facial Animation with Wrinkles. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation '92*, 1-13, 1992.

[38] J. Vince. *Essential Computer Animation Fast.* Springer, 2000.

[39] P. Volino and N. M. Thalmann. Fast Geometrical Wrinkles on Animated Surfaces. In *Proceeding of WSCG '99*, 1999.

[40] K. Waters. A Muscle Model for Animating Three-Dimensional Facial Expression. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21, 4, 17-24, 1987.

[41] A. Witkin and D. Baraff. Physically Based Modeling: Principles and Practice. *SIGGRAPH 97 Course Notes*, 1997.

[42] Y. Wu, N. M. Thalmann, and D. Thalmann. A Plastic-visco-elastic Model for Wrinkles in Facial Animation and Skin Aging. In *Proceedings Pacific Graphics '94*, 201-213, 1994.

[43] Y. Wu, P. Kalra, and N. M. Thalmann. Simulation of Static and Dynamic Wrinkles of Skin. In *Proceedings of Computer Animation '96*, 90-97, 1996.

[44] Y. Wu, P. Kalra, and N. M. Thalmann. Physically-Based Wrinkle Simulation & Skin Rendering. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation '97*, 69-79, 1997.

[45] Y. Wu, P. Kalra, L. Moccozet, and N. M. Thalmann. Simulating Wrinkles and Skin Aging. *The Visual Computer*, 15, 4, 183-198, 1999.