Real-Time Hair Simulation with Segment-Based Head Collision

Eduardo Poyart and Petros Faloutsos

University of California, Los Angeles

Abstract. This work presents a simple, stable and fast hair simulation system for interactive graphics applications whose CPU budget for hair simulation is small. Our main contribution is a hair-head collision method that has very little CPU cost and simulates the volumetric effect of hair resting on top of each other without the need for hair-hair collision. We also present a simulation-based hair styling method for end-users. This easy to use method produces hair styles for the whole head, and it is particularly suitable for abstract and exotic styles. It is applicable for video games in which avatars can be customized by the user.

Keywords: hair simulation, character animation, physics-based motion

1 Introduction

Animating hair in real time remains a hard task in computer graphics. An average human head has in the order of 100,000 highly deformable strands of hair that constantly interact with their neighbors and the scalp. Furthermore, real-time applications such as video games often display many characters at the same time, with hair motion being a secondary effect to other more important tasks that also require computational resources. This leaves very small CPU and GPU budgets for hair simulation, which means that a practical hair simulation system must be extremely efficient.

To address this challenge, many real-time applications trade realism for performance. The most realistic method of animating and rendering hair in real time is rarely the most desirable one, due to those performance constraints. Making simplifications that produce a pleasant result, albeit with some physical inaccuracies, can be very useful if the results have good real-time performance.

A good candidate for simplification or elimination is hair-hair collision, which is generally the most costly operation in hair simulation. We can assume the application doesn't require close-up shots and it's difficult or nearly impossible to notice hair crossing each other. However, one side effect remains: hair strands lie flat at the head surface. This causes a loss of hair volume, and also bad interpenetrating artifacts when common texturing and shading methods are used.

The main contribution of this work is a novel hair-head collision technique that solves the problem of hair strands resting on top of each other without the need for hair-hair collision. This technique is very fast and produces believable

results, and differently from previous work, it doesn't produce artifacts when the head tilts.

As a secondary result, we present a method to allow an end-user such as a video game player to quickly and interactively create exotic and artistic hairstyles. Hair styling is traditionally a complex and time-consuming process, which involves editing meshes or other parameters in a specialized tool. A system for creating entertaining hair styles quickly and easily can be applied, for example, in video games in which the player creates his/her own characters.

It is important to notice that in this work we focus exclusively on the motion of the hair; we consider rendering a separate problem that we will address in the future.

The remainder of this document is organized as follows. On Section 2, we analyze related work in hair simulation and hairstyle modeling. Section 3 shows how we assign hair to the head. Sections 4 describes the simulation and collision method. Section 5 describes the main aspect of our contribution, *segment-based head collision*. Section 6 discusses how our real-time simulator can be used for hair styling, using a force-field-based virtual combing method. The last section summarize our results and describes future work.

2 Related Work

Since the pioneering work of Terzopoulos et al. [17], a lot of work has been done on deformable models and hair. A very complete survey of hair styling, simulation and rendering can be found in [20]. In the remainder of this section, we only discuss the most relevant prior work.

Chang et al. [4] introduced the concept of *guide hair*, a small number of hair strands which undergo physical simulation and guide the rest of the hair. We use this concept in our work.

Real-time results were achieved by some animation techniques, such as [16], which uses the GPU. Our technique, in contrast, uses only the CPU. In some cases this may be desirable – if the GPU is free from the physical simulation, it can be used to render hair using a more complex algorithm in real time.

Hair strands can be simulated as springs and masses [14] or as more complex models, including rigid bodies connected by soft constraints [6]. Springs are revisited by Plante et al. [13], in which their stretchiness is used to model wisps of curly and almost-straight hair, and by Selle et al. [15], in which the authors model a complex arrangement of springs capable of modeling straight and curly hair, but with non-real-time performance.

To achieve real-time performance, Lee and Ko [9] chose not to model hair-hair collision. They obtain hair volume by colliding the hair with different head shells based on pore latitude on the head. Their method suffers from artifacts if the head tilts too much (e.g. looking down). Our method, in contrast, modulates the head shells based on hair segment, instead of pore latitude. We show that our approach achieves the same volumetric results, doesn't suffer from artifacts when the head tilts, and gives the added benefit of allowing spiky hair modeling.

There are many other possible modeling techniques, of various degrees of complexity. Some examples are: loosely connected particles distributed inside the hair volume [1], simulating hair on a continuum ([7], [8], [10]), or trigonal prisms with limited degrees of freedom based on a pendulum model [5].

Hair styling based on meshes was developed by [21]. In this work, an interactive system is used to allow an artist to model hair styles by performing usual mesh editing operations. This model is very flexible, however it is also time consuming. Our method allows almost instant generation of artistic and exotic hair styles, albeit without a lot of that flexibility and control. Also in the area of hair styling, [19] developed automated hair style generation by example, which is useful for rendering large amounts of unique but similar hair styles.

3 Head parametrization

We parametrize the head and randomly assign hair roots by using the Lambert azimuthal projection. One of its important properties is area preservation. It provided good results in distributing hair over the hair. The procedure is described as follows. A random point in the plane xy, uniformly distributed in the circle of radius $\sqrt{2}$, is selected. Its coordinates are (X, Y). This 2D point is stored in the data structure for the strand, and it is also projected into the 3D sphere using the Lambert azimuthal projection, as follows:

$$(x, y, z) = \left(\sqrt{1 - \frac{X^2 + Y^2}{4}} X, \sqrt{1 - \frac{X^2 + Y^2}{4}} Y, -1 + \frac{X^2 + Y^2}{2} \right) \quad (1)$$

Since the radius of the originating circle is $\sqrt{2}$, the projection will occupy the lower hemisphere of the head. The vertex position is then transformed to the upper hemisphere, and rotated by 30 degrees towards the back of the head, which approximates real-life hair distribution. Note that the generation process is separated from the animation process. A better, more accurate hair distribution can be envisioned without any penalty for the run-time system.

4 Hair Animation

In the run-time system, the hair is modeled as masses and springs. Each node has a mass, and springs connects the nodes. The root nodes lie at the head surface. One hair strand is a linear connection of nodes and springs in between them, starting from the root node and extending until the last node. All hairs in the head have the same number of nodes.

Two main reasons led us to choose masses and springs as a modeling method. First, we intended to achieve real-time performance, and by using springs and masses combined with forward Euler integration we achieved results that are computationally very fast. Second, it is an easy method to implement, which

has a positive impact on its applicability in current real-time systems such as video-games.

The run-time physical simulation system perform collision detection and resolution between hair and head, computes forces applied to the nodes due to the springs and gravity, performs integration, and applies error correction. These steps will be explained on this section. We opted not to simulate hair-hair collision to avoid the associated computational cost; instead we focus on techniques to obtain good results without it.

The following pseudo-code shows the hair simulation algorithm.

```
For every simulation step:
    Collide_with_sphere()
    Assign_spring_forces()
    Euler_integration()
    Apply_ERP()

Procedure Collide_with_sphere():
    For every hair strand:
        For every node (except the root):
            If node is inside sphere:
                Correct position
                Assign force to node

Procedure Assign_spring_forces():
    For every hair strand:
        For every segment:
            Add spring force to both nodes

Procedure Euler_integration():
    For every hair strand:
        For every node (except the root):
            Compute acceleration according to force
            Compute velocity according to acceleration
            Compute position according to velocity

Procedure Apply_ERP():
    For every hair strand:
        For every node (except the root):
            Update node position based on ERP
```

4.1 Head Collision

Collision between the hair and the head is performed by sphere collision. If the distance between a node and the center of the head is smaller than r, this node has collided with the head. The center of the sphere and radius r have to be fine-tuned so as to align the sphere as close as possible with the head model

being used for rendering. An ellipse matches human heads more closely. Each node can be transformed by a translation and a non-uniform scaling matrix, such that the collision ellipse becomes a sphere. With that, the usual sphere collision detection can be easily performed.

Once we find a node that penetrates the sphere or ellipse, we need to apply a correction force, which is proportional to the amount of penetration. But before applying this force, we apply a position correction, which will be discussed in Section 4.4. The force is computed using Hooke's law, with a different spring constant ($head\_k_s$) than the one used on the hair segments (k_s). The force is applied to a force accumulator array which has one 3D vector per node per strand. This array is previously set to zero at the beginning of each simulation step.

4.2 Spring Forces

The next step is to add spring forces to the force accumulator array. The forces applied follow Hooke's law:

$$f = -k_s x - k_d v \tag{2}$$

where k_s and k_d are the spring and damping constants, x is the spring extension from the rest length, and v is the current velocity of the node. All segments have the same rest length.

4.3 Euler Integration

The forward Euler integrator simply takes in the time step value and the array of forces applied to nodes, and computes acceleration, velocity and position for each node, as described in the pseudocode.

Euler integration is subject to instabilities when the springs are too stiff, when the time step is too big, or when the masses are too light. We used a fixed time step of 1/60 second. With this time step, we fine tuned the mass and k_s in order to have stability. This resulted in a low spring constant, and the system suffered from loose springs. Instead of replacing the integration method with a more expensive one, we solved this problem with the technique explained below.

4.4 Error Reduction Parameter (ERP)

With a spring-mass system for hair strands, the length of each hair segment is not guaranteed to be constant, especially when k_s is low. In practice, this means that each spring will extend under the weight of the nodes and the hair will stretch, and worse, it will bounce up and down. This is a major cause of unnatural results for hair modeled this way. If k_s is increased to achieve stiffer springs, the system becomes unstable unless k_d is increased as well.

One solution would be to use a rigid body model for each segment and solving a system of equations for each hair strand. This was not implemented due to the

computational cost that would result. The algorithm would no longer be linear on the number of segments per hair, unless a more advanced algorithm such as Featherstone [11] was used.

Another solution is to implement an error reduction parameter (ERP), similar to the one implemented in Open Dynamics Engine [12]. ERP is simply a scalar value between 0 and 1 that defines how much of the length error will be corrected at each time step. The correction is performed as a change in position of each node, along the length of the spring, towards the correct length. If ERP=0, there is no correction. If ERP=1, there is full correction: each node will be moved so that the length is fully corrected in one time step.

Why not always use ERP=1, correcting by the full amount every time step? Performing error reduction in this fashion has the practical effect of introducing damping on the system. Tests made with ERP=1 showed that the system became too damped, so smaller values had to be used. Common values for all parameters are presented on Table 1. With these values, k_d introduces virtually no damping while ERP is set to the highest value possible that would not result in too much damping – "too much damping" being an empirically observed behavior. This gave us the best possible constraint on hair length. It is important to notice that this achieves a better constraint/damping ratio than what would be achieved by increasing k_s and k_d alone without the use of ERP.

Table 1. Simulation parameters that provided realistic and stable results.

Parameter	Value
head radius	1
number of segments	10
hair length	0.1 to 2.0
k_s	15
k_d	0.0001
mass	0.001
erp	0.6
$head\_erp$	0.5
$head\_k_s$	5
$sphere\_distance\_factor$	0.02

4.5 Guide Hair

We employ the guide hair technique [4] in order to get as close as possible to 100,000 hair in real time. We physically simulate only a small amount of hair and interpolate the nodes of the remaining hair. We found that physically simulating 2000 guide hair and rendering 20,000 to 40,000 hair typically give adequate, interactive performance.

5 Segment-Based Head Collision

As described so far, the system is real-time and stable, and maintains an almost constant length for simulated hair strand. However, a major drawback is that the hair resting on top of the head is very flat, with no volume being formed due to layers of hair lying on top of each other. In this section we describe previous efforts to overcome this problem, and then our technique to do the same, and we compare it to a more expensive method of hair-hair collision.

Lee and Ko [9] achieve volume by increasing the head collision radius based on pore latitude on the head. The head collision sphere is increased the most when the pores are close to the top of the head. This leads to a result where layers of hair whose pores are higher on the head rest on top of other layers. The biggest drawback of this technique is that, if the head tilts, the order is inverted and the result is unnatural, with hair strands crossing each other. The more the head tilts, the more this crossing over happens.

Let us consider the fact that our hair is a sequence of straight line segments. Between these segments there are nodes, which are indexed, starting with 0, which is the root of the hair strand on the head, and ending with the maximum number of nodes in a strand. In order to maintain layers of hair on the head, the ordering of nodes can be used to slightly increase the sphere radius step by step. That is, the further away from the root a node is, the further away from the head will be its actual collision surface. Fig. 1 depicts this idea. A parameter called sphere distance factor (sdf) represents the increase in radius that is in effect at the last node in the hair strand. Therefore, the per-node increase is sdf/n, n being the number of nodes. Parametrizing sdf this way allows the user to change the number of nodes, e.g. to improve performance, and have consistent visual results, without having to change sdf as well to compensate.

We compared this technique with hair-hair collision. In our hair-hair collision implementation, the main computational primitives used were line-line distance and point-line distance functions. Both penalty forces and position displacement were attempted. We had oscillations in the system, especially at the top of the head. Furthermore, with nothing but guide hair being simulated, there was not enough guide hair to achieve volume. The interpolated hair still appears flat. Chang et al., in [4], attacked this problem by adding triangle strips between guide hair, with an added performance penalty.

Even though hair-hair collision is more general, our approach of index-based head distance comes virtually for free and solves the problem of hair strands resting on top of each other. The flatness of the head is avoided, and hair strands no longer crisscross each other when lying on the head. It works even if only a small number of guide hair are simulated.

As an added bonus, a new hair style can be generated: spiky hair. This can be achieved by making sdf large enough. By making the collision radius grow linearly with the segment index, we had surprisingly interesting results. A high sdf produces great fur simulation. Both spiky hair and fur are depicted on Fig. 2. Other extensions can be devised. The collision radius growth need not be linear. A quadratic increment, for example, makes the hair more spiky. Exotic collision

shapes can also be used. Figure 3 shows the result of a collision sphere whose horizontal cross-section is perturbed by a sine wave. This creates vertical "hills and valleys" which shape the hair. This figure also show how volume is retained with segment-based head collision even if long hair is used.

The accompanying video in http://www.erpoyart.com/research/hair shows a comparison between the previous latitude-based head collision and our index-based head collision. It highlights hair layers crossing each other in the latitude-based method, which doesn't happen with index-based collision.

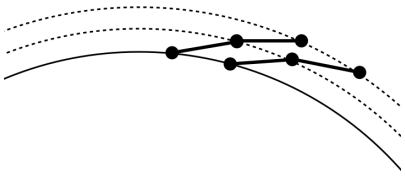

Fig. 1. Segment based collision. The thicker line segments are hair segments; the solid circle represents the head surface where the root nodes lie and the dashed circles are collision spheres for subsequent nodes of the hair.

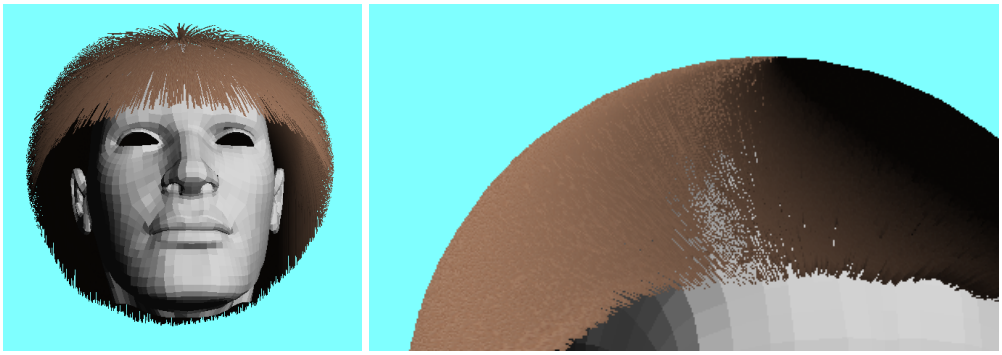

Fig. 2. Spiky hair (left) and fur (right) obtained using segment based collision.

6 Hair Styling

Achieving interactive physically-based hair simulation gives rise to interesting possibilities. One of them is the easy modeling of hair styles by pausing the physical simulation at any point in time and interacting with the hair.

This feature was implemented in our system. The user can interactively rotate and move the head, which will cause the hair to move under physics, and he/she

Fig. 3. Left: An exotic collision shape gives "hills and valleys" to the hair. Right: Volume is retained with long hair.

can pause and continue the simulation at will. Pausing the simulation causes each node to be attached to its current position in head-space through zero-length springs. The attachment points are called anchors. The hair is still simulated and it responds to head movements and other agents like wind, as if the character applied gel to fix his hairstyle.

We also allow the user to perform "virtual combing". Mouse drag operations are transformed from screen coordinates to world coordinates, lying on a plane parallel to the screen and just in front of the head. This plane intersects with many of the hair strands closest to the user. The movement direction, V, is a vector pointing from the previous to the current projection point. In the vicinity of the current projection point P, we define a vector field whose magnitudes follow a Gaussian fall-off law:

$$m = e^{(-kd^2)} \tag{3}$$

where d is the distance between P and a point in the vector field, and k is an empirically determined constant that defines how big the area of actuation is. The scalar m is computed for each node in each strand of hair. If it lies above a certain threshold (we used 0.1), it scales the vector V and the resulting vector modifies the node position.

With a little practice, the user can quickly figure out many ways to obtain hair styles with this method. In Fig. 4, a few examples are shown in which only head movements were applied, and in Fig. 5 head movements and virtual combing were applied together.

Due to the simplicity and "fun-factor" of this hair styling technique, and due to the fact that it can generate very artistic and abstract hair styles, we envision its immediate use in video game applications, particularly RPGs or massively multiplayer games in which the user creates and customizes his/her character (avatar) before starting the game. This customization nowadays mainly supports the selection of one out of a number of predefined hair styles that can be customized minimally. Such functionality can be found for exampled in

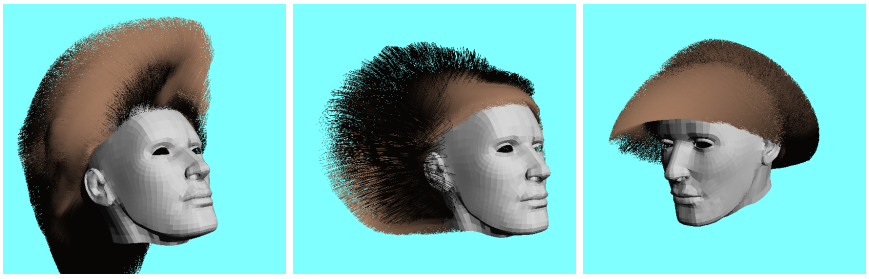

Fig. 4. Examples of different hair styles obtained using the system, without combing.

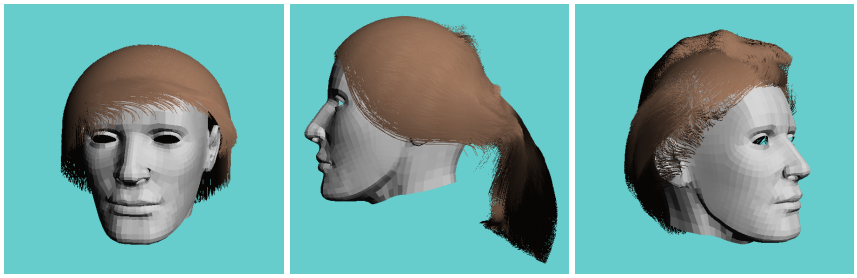

Fig. 5. Hair styles obtained by combing.

the popular games Oblivion [2] and World of Warcraft [3]. Allowing the user to customize a hair style interactively with a physically based system would be something new and potentially entertaining. Even for systems that can't animate hair in real time due to CPU budget constraints, the physically-based system can run during the character customization phase since the CPU is not heavily required at that point (no game logic is running and no other characters are being rendered), and the character can appear in-game with a static rendering of the styled hair created by the user. This would add an element of fun to the game.

7 Results

The present algorithm is very fast and linear in time complexity, simulating around 20,000 hair strands at 30 frames per second, or around 40,000 at 15 frames per second. It produces good visual results and it is fun to interact with. The performance effect of introducing guide hair is shown on Table 2. An even more important indication of performance is the time taken for the physics simulation, including integration and collision of guide hairs and update of slave hairs: around 4.4 ms – only 1/8th of a frame at 30 fps on a single CPU core. We were able to simulate multiple simultaneous heads with hair. On the accompanying video, we show three heads with hair simulated at 30 frames per second (the frame rate is shown at 25 due to the cost of video capture).

The method has a few drawbacks. Since there is no hair-hair collision, there is hair interpenetration. In a real-world application such as a simulation or video-game, care must be taken so that this effect is hidden from the user as much as possible. We noticed that if the character is not being viewed at close-up, hair-hair interpenetration is not noticeable. The modeling system, although easy to use, does not allow for detailed control. It is primarily based on trial and error. Our implementation of virtual combing also does not allow precise fine-tuning; it is geared towards major changes in hair positioning.

Table 2. Effect of guide hair on performance. Results obtained on a 2.5 GHz Intel Core 2 Duo, using only one core.

Simulated (Guide) Hair	Slave Hair	Frame Rate	Physics Sim. Time
20,000	0	19 fps	23 ms
1,000	19,000	33 fps	4.4 ms

8 Conclusion

This work introduced a real-time hair simulation system that is very fast and relatively simple to implement. Its time complexity is linear on the number of hair strands and number of nodes per hair strand, which means the number of strands can be greatly increased, up to the same order of magnitude as the human head, and still maintain interactive frame rates. We achieve volume on the hair, as opposed to a flat hair style, by means of segment-based head collision.

A simple modeling method is also introduced. By simply moving the head, freezing the system and applying virtual combing, the user can quickly get interesting hair styles. This is especially useful for abstract and artistic hair styles. This styling method can be applied to situations such as video games where the player is able to customize an avatar. In that kind of situation the requirement is usually that the modeling process shouldn't be lengthy, complex or detailed; it should be simple and fun to use. The fact that our system runs entirely in real-time with attractive results makes this possible.

There are many avenues to extend this work. We would like to experiment with other integration schemes, such as Verlet integration, and articulated body representations of hair.

The styling system can be extended in many different ways so that the user has better control of the results. A combination of user defined vector-fields and physical interaction would be very interesting to pursue. Furthermore, the results of the user-generated hair styles developed here can be fed into a system such as [19], and variations of the hair styles can be automatically generated. Hair rendering is another aspect that will be addressed in the future, since in the current work we have opted to focus exclusively on modeling and animation.

References

1. Bando, Y., Chen, B.-Y., Nishita, T.: Animating Hair with Loosely Connected Particles. Computer Graphics Forum (Eurographics Proc.), Volume 22, Issue 3 (2003)
2. Bethesda Softworks: The Elder Scrolls: Oblivion (video game title). http://www.elderscrolls.com/games/oblivion_overview.htm (2004)
3. Blizzard: World of Warcraft (video game title). http://www.worldofwarcraft.com/ (2004)
4. Chang, J. T., Jin, J., Yu, Y.: A Practical Model for Hair Mutual Interactions. ACM Transaction on Graphics (Proc. of SIGGRAPH), p.73-80 (2002)
5. Chen, L. H., Saeyon, S., Dohi, H., Hisizuka, M.: A System of 3D hair Style Synthesis based on the Wisp Model, The Visual Computer, Springer Verlag, 15(4), p.159-170 (1999)
6. Choe, B., Choi, M., Ko, H.-S.: Simulating complex hair with robust collision handling. In Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., 153-160 (2005)
7. Hadap. S., Magnenat-Thalmann, N.: Interactive hair styler based on fluid flow. Computer Animation and Simulation, Proc. of the Eleventh Eurographics Workshop (2000)
8. Hadap. S., Magnenat-Thalmann, N.: Modeling dynamic hair as a continuum. In Comp. Graph. Forum (Eurographics Proc.), 329-338 (2001)
9. Lee, D.-W.; Ko, H.-S.: Natural Hairstyle Modeling and Animation. Graphical Models, Volume 63, Number 2, pp. 67-85 (2001)
10. McAdams A., Selle, A., Ward, K., Sifakis, E., Teran, J.: Detail Preserving Continuum Simulation of Straight Hair. ACM Transactions on Graphics (Proc. of SIGGRAPH) 28, 3 (2009)
11. Mirtich, B.: Impulse-based Dynamic Simulation of Rigid Body Systems. Ph. D. Thesis, University of California at Berkeley (1993)
12. ODE – Open Dynamics Engine. http://www.ode.org/
13. Plante, E., Cani, M. P., Poulin, P.: A Layered Wisp Model for Simulating Interactions Inside Long Hair. Proceedings of the Eurographic workshop on Computer animation and simulation, pp. 139-148 (2001)
14. Rosenblum, R. E., Carlson, W. E., Tripp III, E.: Simulating the structure and dynamics of human hair: modelling, rendering and animation. J. Vis. and Comput. Anim. 2, 4, 141-148 (1991)
15. Selle, A., Lentine, M., Fedkiw, R.: A mass spring model for hair simulation. ACM Trans. on Graph. 27, 3, 1-11 (2008)
16. Tariq, S., Bavoli, L.: Real time hair simulation on the GPU. ACM SIGGRAPH talks, session: Let's get physical (2008)
17. Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K.: Elastically deformable models. In Computer Graphics, pp. 205-214 (1987)
18. Volino, P., Magnenat-Thalmann, N.: Animating Complex Hairstyles in Real-Time. Proceedings of the ACM symposium on Virtual reality software and technology, Hong Kong, pp. 41-48 (2004)
19. Wang, L., Yu, Y., Zhou, K., Guo, B.: Example-based hair geometry synthesis. ACM Transactions on Graphics (Proc. of SIGGRAPH) 28, 3, Article 56 (2009)
20. Ward, K., Bertails, F., Kim, T.-Y., Marschner, S. R., Cani, M.-P., Lin, M. C.: A Survey on Hair Modeling: Styling, Simulation, and Rendering. IEEE Transactions on Visualization and Computer Graphics, pp. 213-234 (2007)
21. Yuksel, C., Schaefer, S., Keyser, J.: Hair meshes. Proc. of SIGGRAPH Asia (2009)