Volume 28 (2009), Number 7 pp. 1–8 COMPUTER GRAPHICS *forum*

Chain Shape Matching for Simulating Complex Hairstyles

W. Rungjiratananon[1], Y. Kanamori[2] and T. Nishita[1]

[1]The University of Tokyo, Japan
[2]University of Tsukuba, Japan

Abstract
Animations of hair dynamics greatly enrich the visual attractiveness of human characters. Traditional simulation techniques handle hair as clumps or continuum for efficiency; however, the visual quality is limited because they cannot represent the fine-scale motion of individual hair strands. Although a recent mass-spring approach tackled the problem of simulating the dynamics of every strand of hair, it required a complicated setting of springs and suffered from high computational cost. In this paper, we base the animation of hair on such a fine-scale on Lattice Shape Matching *(LSM), which has been successfully used for simulating deformable objects. Our method regards each strand of hair as a chain of particles, and computes geometrically-derived forces for the chain based on shape matching. Each chain of particles is simulated as an individual strand of hair. Our method can easily handle complex hairstyles such as curly or afro styles in a numerically stable way. While our method is not physically-based, our GPU-based simulator achieves visually-plausible animations consisting of several tens of thousands of hair strands at interactive rates.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Simulating the dynamics of a full head of hair (i.e., typically one hundred thousand hair strands) has, for a long time, been a challenging task in computer graphics. The difficulty mainly stems from the computational cost and numerical stability, especially in the interactions between individual strands of hair, which are essential for animation. Other important concerns for hair simulation include how to model each strand to represent various hairstyles; many recent techniques handle hair as clumps or a continuum to avoid the computational issues and focus on specific hair styles such as straight only [BNC03, Osh07, TB08] or straight plus curly hairstyles [VMT06, BAC*06]. Recently, an extended mass-spring model [SLF08] undertook to simulate the dynamics of up to ten thousand individual hair strands. However, this required a complicated configuration for the spring structure and suffered from high computational cost.

In this paper, we undertake fine-scale animation of a large number of individual hair strands based on *Lattice Shape Matching* (LSM) [RJ07]. LSM has been successfully used for simulating deformable objects because it is simple, fast and numerically stable. While LSM assumes that a deformable object can be approximated by a set of particles aligned in a lattice, we represent a hair strand as a chain of

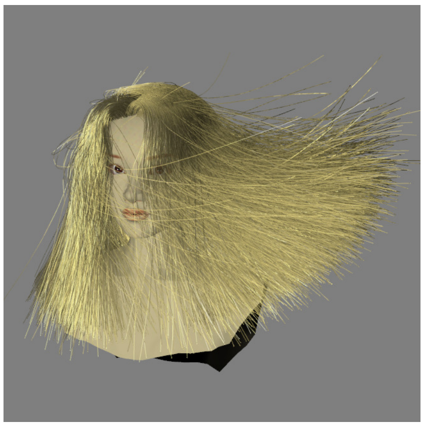
Figure 1: *Animating 10k straight strands of hair (16 particles per strand, about 12 fps on the GPU), with the dynamics of each strand of hair.*

Table 1: *Advantages of our method against previous methods for hair strand dynamics.*

	Rigid chain [HMT01]	**Super helices** [BAC*06]	**Mass-spring** [SLF08]	**Our method**
Stability	conditional	conditional	unconditional	unconditional
Hairstyle	straight only	complex configurations	complex configurations	easy
Constraint	tricky	tricky	easy	easy
Performance	off-line	off-line	off-line	interactive

particles; therefore we call our method *Chain Shape Matching* (CSM). CSM is much simpler than LSM because it only considers deformations along a single chain. We can immediately use the particles' positions in the original shape to create a chain structure, which greatly benefits the design process. While a hair strand should not be stretched, the original LSM was not concerned with stretching because it targeted deformation only. However, we successfully integrated a stretching constraint into the LSM framework.

Our proposed model is not physically-based, targeting interactive applications including games. Our method can be implemented entirely on the GPU and achieve interactive performance for up to several tens of thousands of hair strands (Figure 1).

2. Previous Work

The early research on hair simulation handled the dynamics of individual strands of hair using the mass-spring system [RCT91] and projective dynamics [AUK92]. However, these methods ignore hair-hair interactions, curliness and torsion. Hadap and Magnenat [HMT01] use a rigid multibody serial chain to represent a hair strand which can capture the torsion effect, but still ignore curliness. Bertails et al. [BAC*06] introduced a mechanical model called *super-helixes* based on Kirchhoff's theory to represent a strand of hair. The model can capture bending, torsion, non-stretching and the curliness behavior of hair at high expense of computational cost.

For computational efficiency, several techniques regard hair as disjoint groups [DTKT93, PCP02] or a continuum [HMT01, BNC03, VMT06], limiting the degrees of freedom (DOFs) of the hair motion; although human hair has a collective tendency, a high number of DOFs is required to represent the fine-scale motion of hair blown about by the wind. Please refer to the survey [WBK*07] for advances on hair modeling, styling, simulation and rendering.

Recently, mass-spring systems have commonly been used for simulating hair dynamics. Integrations in mass-spring systems are performed using explicit or implicit schemes. Explicit schemes are often preferred due to the low computational cost and ease of implementation. However, for stable simulation, the time step in an explicit scheme should be inversely proportional to the square root of the spring constant (the Courant condition). As a result, highly stiff hair is difficult to simulate at an interactive rate when using an explicit scheme. Implicit schemes can solve the stability problem, with higher computational cost. Selle et al. [SLF08] proposed the use of additional altitude springs to simulate the complex behavior of human hair and semi-implicit springs to solve the stability problem. However, this is not suitable for stylized hair in interactive applications due to the expensive cost and complex configuration of the springs.

As for hair-hair interactions, most of the previous methods only consider collisions occurring on guide hair strands [CCK05, BAC*06] and possibly miss collisions between interpolated hair strands when the guide strands do not collide. There has been little research done that takes full hair-hair interactions into consideration, except for the time-consuming bounding box hierarchy used in [SLF08] and the hybrid technique of Eulerian and Lagrangian methods introduced in [MSW*09]. Tariq and Bavoil [TB08] introduced a fast technique for inter-hair collisions using a hair density field. All the hair strands are voxelized into a grid then repulsive forces are applied to hair particles in high density areas. However, their technique only affects volume preservation of hair. The same limitation can be seen in the continuum methods [HMT01, BNC03, VMT06].

Unlike most of the previous works, our method is easy to implement, can easily handle complex hairstyles, is interactive and numerically stable even if the hair is very stiff. Also, our method handles collisions between individual hair strands. This is accomplished by using particle-particle collision detection. The advantages of our method for simulating hair strand dynamics against those described in previous works are summarized in Table 1.

3. Hair Simulation Method

Here we briefly explain Lattice Shape Matching (LSM) [RJ07] because it forms the basis of our method. Then we describe how to model hair using our Chain Shape Matching (CSM). While hair strands, in actual fact, do not stretch, they are stretched in LSM. We also introduce strain limiting for CSM. Finally, we describe the hair-hair interaction algorithm in our model.

3.1. Lattice Shape Matching (LSM)

LSM is an extension of the shape matching method [MHTG05]. The main advantages of the shape matching method are unconditional stability and high controllability due to its geometrically-motivated computation. Optimally-transformed positions are computed first, and then particles are moved towards those positions. Since it guarantees that all particles are updated towards the appropriate positions, the overshooting problem that occurs in explicit integration schemes is eliminated. This technique is later generalized as *position based dynamics* [MHHR07], which can be applied to general simulation systems.

In LSM, the particles are grouped into multiple-overlapping cubical regions (Figure 2b). The region half-width value w $(w = 1, 2, 3, \cdots)$ corresponds to the stiffness of the object. The positions of particles are updated as follows. First, they are moved independently according to the external forces. Next, for each region, LSM computes an optimal rigid transformation (i.e., rotation and translation) based on shape matching [MHTG05]. The rigidly-transformed positions of the particles are called *goal positions*. The goal position $\mathbf{g}_i$ of particle i is weighed in the overlapping regions by particle per-region mass $\widetilde{m}_i = \frac{m_i}{N_r}$, where m_i is the mass of particle i and N_r is the total number of regions that the particle i belongs to. The goal position of particle i is computed as follows.

$$\mathbf{g}_i = \frac{1}{N_r} \sum_{r \in \mathcal{R}_i} (\mathbf{R}_r(\mathbf{x}_i^0 - \mathbf{x}_{cm,r}^0) + \mathbf{x}_{cm,r}), \tag{1}$$

where $\mathcal{R}_i$ is a set of regions that particle i belongs to, $\mathbf{x}_i^0$ is the original position, $\mathbf{x}_{cm,r}^0$ is the center of mass of the original shape of the region, $\mathbf{x}_{cm,r}$ and $\mathbf{R_r}$ are the optimal translation and rotation for region r. Finally, for each particle, the velocity is computed toward the goal position.

$$\mathbf{v}_i(t+dt) = \mathbf{v}_i(t) + \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{dt} + dt\frac{\mathbf{f}_{i,ext}(\mathbf{x}_i, t)}{m_i}, \tag{2}$$

$$\mathbf{x}_i(t+dt) = \mathbf{x}_i(t) + dt\mathbf{v}_i(t+dt), \tag{3}$$

where dt is a time step, $\mathbf{v}_i$ is the velocity and $\mathbf{f}_{i,ext}$ the external force.

3.2. Hair Simulation Using Chain Shape Matching (CSM)

In our method, a hair strand is represented by a chain of particles grouped into multiple overlapping chain regions (Figure 2d). Each particle i is associated with a chain region $R_i \in \mathcal{R}_i$ that centers the particle i and contains adjacent particles within the region half-width w. Each chain region uses the same shape matching method as used in LSM (Figure 3). Therefore we call our algorithm Chain Shape Matching (CSM). Algorithm 1 describes the pseudocode of our algorithm.

Note that the rotation matrix $\mathbf{R}$ can be inverted, i.e., it can involve reflection. Reflection causes flipping of the input shape, and therefore should be avoided for ordinary shape matching [MHTG05, RJ07]; however, it has little influence

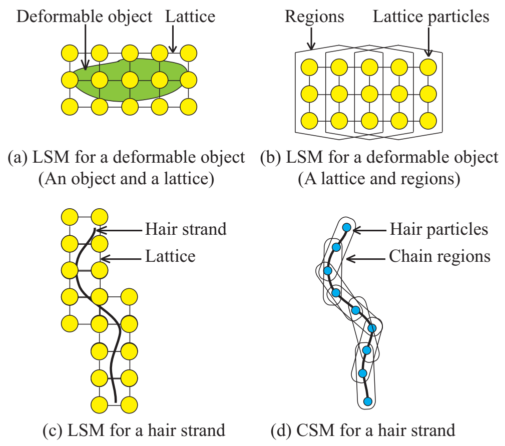
Deformable object
Lattice
Regions
Lattice particles
(a) LSM for a deformable object (An object and a lattice)
(b) LSM for a deformable object (A lattice and regions)
Hair strand
Lattice
Hair particles
Chain regions
(c) LSM for a hair strand
(d) CSM for a hair strand

Figure 2: *Illustrations for* Lattice Shape Matching *(LSM) and* Chain Shape Matching *(CSM).*

Algorithm 1 Pseudocode of CSM algorithm.

```
 1: for all particles i do
 2:   initialize original position $\mathbf{x}_i^0$, $\mathbf{x}_i \leftarrow \mathbf{x}_i^0$
 3: end for
 4: loop
 5:   BucketGeneration() // Sections 3.4 and 4.1
 6:   for all particles i do
 7:     $\mathbf{f}_{i,ext} \leftarrow$ ComputeCollisionForce() + $\mathbf{f}_{gravity}$
 8:   end for
 9:   for all particles i do
10:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + dt\frac{\mathbf{f}_{i,ext}}{m_i}$
11:     $\mathbf{x}_i \leftarrow \mathbf{x}_i + dt\mathbf{v}_i$
12:   end for
13:   for all chain regions $R_i$ do
14:     $\mathbf{x}_{cm} \leftarrow$ ComputeOptimalTranslation()
15:     $\mathbf{R} \leftarrow$ ComputeOptimalRotation()
16:   end for
17:   for all particles i do
18:     $\mathbf{g}_i \leftarrow$ ComputeGoalPosition() // Eq.1
19:   end for
20:   for all particles i do
21:     $\mathbf{g}_i \leftarrow$ StrainLimiting() // Section 3.3
22:     $\mathbf{x}_i \leftarrow \mathbf{g}_i$
23:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\mathbf{g}_i - \mathbf{x}_i}{dt}$
24:   end for
25: end loop
```

in our case because we employ one-dimensional shapes, and thus can be ignored.

In CSM, we can design complex hairstyles at the strand-level as follows. The shape of the strand can be defined by the original particle positions, since these original positions define the shape of the strand at rest (Figure 4). The root of each strand is fixed by several particles which are

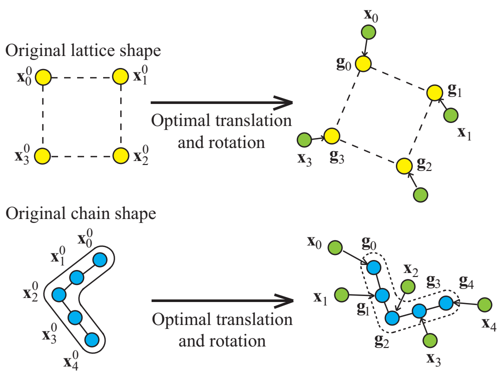

Figure 3: $\mathbf{x}_i^0$ *is the original position,* $\mathbf{x}_i$ *is the position updated by external forces and* $\mathbf{g}_i$ *is the goal position of particle i.*

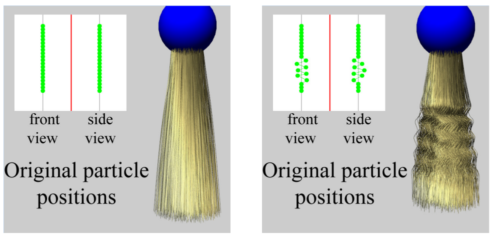

Figure 4: *The original particle positions define the shapes of the hair strands.*

constrained on the head. The number and direction of the constrained particles partially determine the behavior of the strand. The stiffness of the strand is defined by the chain region half-width w (Figure 5), which can be partially modified to generate complex hairstyles; e.g., soft straight hair near the root and stiff curly hair near the tip of the hair strand.

Regarding the stiffness control of hair strands, one might consider the use of parameters $\alpha, \beta \in [0,1]$, as presented in the original shape matching paper [MHTG05]; α controls the tendency that goal positions are moved towards rigidly-transformed positions, and β allows goal positions to undergo a linear transformation (see [MHTG05] for more details). While α and β can control the stiffness independently of w, we simply fix $\alpha = 1$ and $\beta = 0$ according to the LSM paper [RJ07], taking into account that α and β can make a region softer but not stiffer. Although reducing the number of particles makes a region stiffer, it also reduces the hair strand's DOFs required especially for complex hairstyles. Nevertheless, the use of α and β in conjunction with w might benefit to advanced stiffness control, which is left as future work; a study on the relationship between β and w can be found in [OKN09].

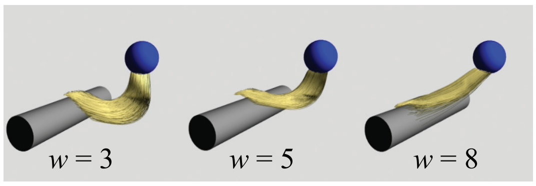

Figure 5: *A bunch of 1,000 strands is dragged over a cylinder, with different chain region half-width w. Small w makes the strands softer while large w stiffer.*

3.3. Strain Limiting

In this subsection, we describe how to limit the stretching of hair. Techniques for constraining stretching were originally proposed in research on cloth simulation, known as *strain limiting*. Most of the cloth simulations use an elastic system to model cloth such as popular mass-spring systems. As a drawback, the cloth becomes very elastic and looks unnatural. Provot et al. [Pro95] therefore modified positions by checking the length of each segment (a spring in the mass-spring systems) in the cloth and then adjusting the length of stretched segments. However, modifying one segment causes stretching of other connected segments. Although iterative modification can be applied, convergence is not guaranteed. Bridson et al. [BFA02] and Selle et al. [SLF08] modified velocity instead of position. In our framework, however, goal positions are computed first and then the velocities are computed so that particles are moved towards the goal positions (Section 3.1). Therefore, modifying velocity is not appropriate for our CSM framework (position based framework).

To constrain stretching, we modify the positions of stretched segments in hair strands after the shape matching process (Figure 6a). Since the root of a hair strand is always attached to the head, we adjust only the near-tip particle (the endpoint near the tip of the strand) of a stretched segment to the non-stretched position. This adjustment process takes place from the root to the tip of the strand, therefore an iterative adjustment is not required.

A simple option for strain limiting is to shrink segments individually and then connect them together (Figure 6b); however, our experiments yield unnatural results. Instead, we move each near-tip particle along the direction of the previous segment that is already adjusted (Figure 6c). As a result, each hair strand becomes a smooth curve from the root to the tip (Figure 6d). Note that just moving particles to the non-stretched positions leads to the loss of linear and angular momenta. To conserve the momenta, our method modifies velocity, similarly to [MHHR07].

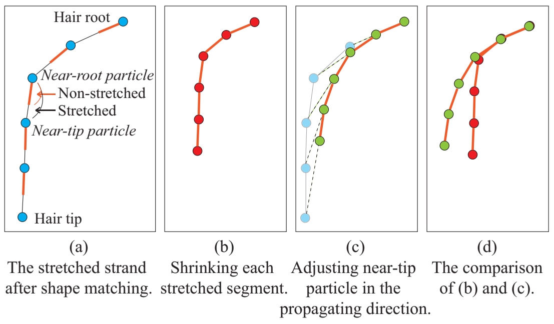
Hair root
Near-root particle
Non-stretched
Stretched
Near-tip particle
Hair tip
(a)
The stretched strand after shape matching.
(b)
Shrinking each stretched segment.
(c)
Adjusting near-tip particle in the propagating direction.
(d)
The comparison of (b) and (c).

Figure 6: *Comparison of stretched particles adjustment.*

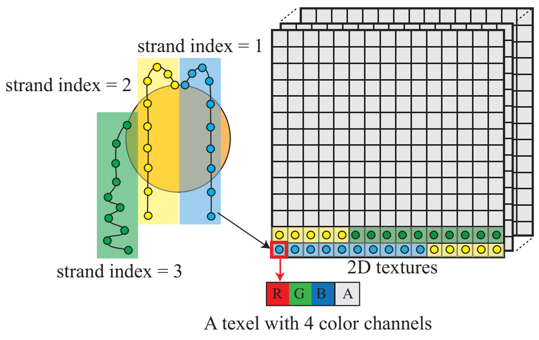

Figure 7: *Physical values of particles are stored in textures (up to four values per texel). Particles of each hair strand are distinguished by the strand index. These are shown by different colors in this figure.*

3.4. Collision Detection and Response

For hair-hair and hair-head interactions, we employ simple sphere-sphere collision detection that can be easily implemented on the GPU. Particles of hair strands required in CSM are used for collision detection. Note that, to handle collisions more accurately, we can increase the number of particles independently of those used in CSM, while our implementation uses identical particles. Particle-particle collisions are detected using a uniform grid of voxels that store the indices of particles. The pair particles colliding are found from the neighboring $3\times3\times3$ voxels, and the penalty forces are added to them. For hair-head interactions, the head model is represented by a set of large spheres, and collisions with each particle are calculated.

4. GPU Implementation

For the GPU computation, the physical values of particles are stored in 2D textures in the GPU memory. We use 32bit floating-point RGBA textures, and thus each texel can store up to four values for a particle. Our implementation uses ten textures, i.e., two position textures, two velocity textures, one property texture (region half-width w, particle index, strand index and limiting length), one original position texture, one texture for optimal translation and three textures for optimal rotation (3×3 matrix, 9 components). Strand indices (see Figure 7) are required to distinguish strands because values for all particles are stored in the same textures, and each strand can contain a different number of particles. In addition, our implementation requires one bucket texture, where each texel represents a voxel that stores particle indices for nearest-neighbor searches in the collision detection process.

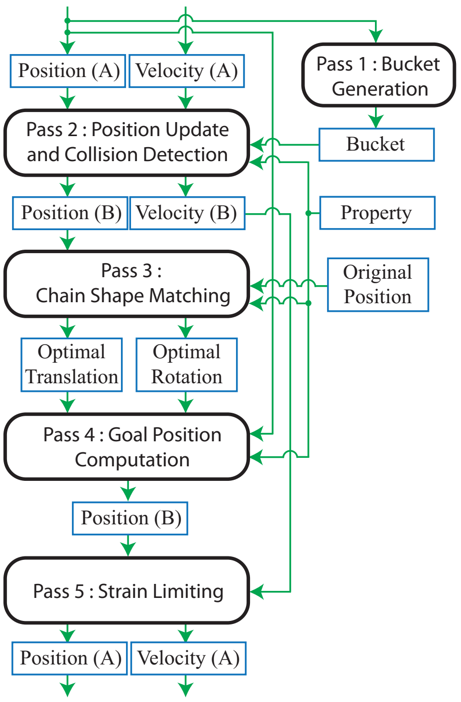

Figure 8: *Simulation flow of a single step on the GPU. Blue rectangles represent the texture data, black rounded rectangles represent operations and green directed line segments represent the flow of data.*

4.1. Simulation Step

In each simulation time step, five passes of the GPU computation are assigned (see the black rounded rectangles in Figure 8). The detail of each GPU pass is explained together with corresponding line numbers of Algorithm 1 as follows:

Passes 1 and 2 : Position update and collision detection (lines 5-12) First, the bucket texture is generated for nearest-neighbor searches using a GPGPU technique [Har07]. The velocity of each particle is updated according to external forces such as gravity and penalty forces calculated in the collision detection process (Section 3.4). Then, the position of each particle is updated according to the updated velocity and the time step.

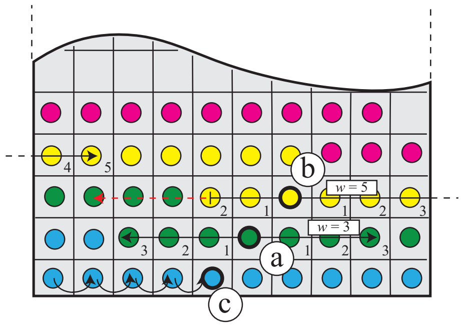

Figure 9: *Connectivity information of hair particles on a texture. (a) The region half-width* $w = 3$*. The particles in the region can be found in the adjacent texels. (b) The region half-width* $w = 5$*. All five adjacent particles in the right half can be found. The left half has only two adjacent particles in a strand. (c) Access pattern for strain limiting. The non-stretched position is computed by tracing the length of each segment from the root particle.*

Pass 3 : Chain shape matching (lines 13-16) The optimal rigid transformation of each chain region is computed from the particle positions in the region. Hair particles are stored sequentially in a texture. See Figure 9 for an example of the texture layout. Particle a has a half-width size $w = 3$. Particles contained in the region can be found in three texels to the left and right hand sides. In the case of particle b with a half-width size $w = 5$, five adjacent particles can be found on the right hand side, while the last two particles are found in the next row by computing their addresses with the texture width. However, the last three particles on the left hand side have a different strand index. Therefore, only two adjacent particles can be found. The region doesn't have to contain a maximal number of particles (eleven in case of $w = 5$).

Pass 4 : Goal position computation (lines 17-19) After the optimal translation and rotation computation, the goal position of each particle is computed by averaging the goal positions of overlapping regions (Equation 1). This process can be computed in a similar way to the chain shape matching pass (Pass 3). Instead of particle positions, the computed optimal translation and rotation of the overlapping regions can also be read from the adjacent texels.

Pass 5 : Strain Limiting (lines 20-24) The final pass is the strain liming process (Section 3.3). The non-stretched position of each particle can be computed from the length starting from the root particle of the strand to the particle (see particle c in Figure 9). After updating particle positions, the velocities are also updated in this pass.

5. Results

The prototype implementation was written in C++, using OpenGL and GLSL. All experiments were conducted on a PC with an Intel Core 2 Quad 3.00GHz, 2GB RAM and an NVIDIA GeForce GTX 280 graphics card. The structure of hair is rendered as connected line segments between particles, and the visual quality is enhanced by Catmull-Rom splines on the GPU using the instanced tessellation technique [BS05]. As for the shading and self shadowing of hair, we used the Kajiya-Kay shading model and Deep Opacity Maps [YK08], respectively. All simulation and rendering were entirely conducted on the GPU. The frame rates in this paper include both simulation and rendering. We perform only a single simulation step per frame. Figure 10 shows the result of hair simulation with and without strain limiting described in Section 3.3. The hair strands are stretched due to the external forces without strain limiting. Figure 11 shows the result with different numbers of strands on the head. With more strands the visual quality is increased.

Figure 13 shows animation sequences of straight, curly and complex hairstyles flowing in the wind. Each strand of the complex hairstyle is straight around the top and curly around the bottom. Each scene consists of 10k strands (160k particles), 10k strands (580k particles) and 17k strands (764k particles), respectively. The breakdown computational time used in each GPU pass and rendering for each result is shown in Figure 12. The simulation and rendering speeds of each sequence are 12, 7 and 4 fps, respectively.

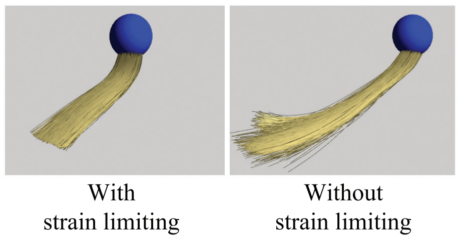

Figure 10: *Moving bunch of hair strands with (left) and without (right) strain limiting.*

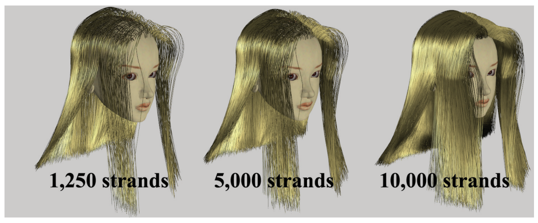

Figure 11: *Increasing hair strands greatly improves the visual quality. From left to right, there are 1.25k, 5k and 10k hair strands on the head.*

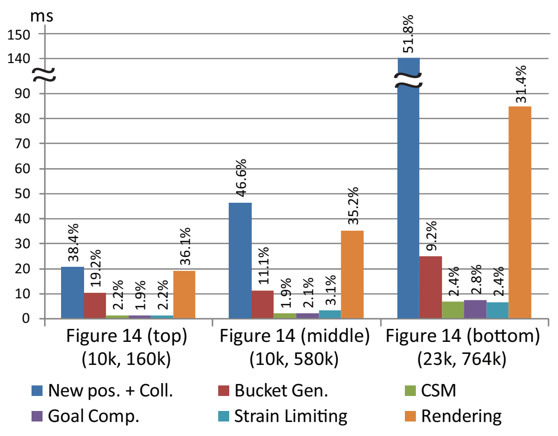

Figure 12: *The computational time in milliseconds used in each GPU pass and rendering. The detail of each GPU pass is described in Section 4.1. The numbers of hair strands and particles used in each result are shown as (no. of strands, no. of particles).*

6. Conclusions

We have presented a simple model for simulating hair based on shape matching. Our method can achieve visually-plausible animations with complex hairstyles in a numerically stable way, even for highly stiff and curly hair like an afro. We have also demonstrated that a GPU-based simulator can achieve interactive performance up to several ten thousand hair strands.

There are some limitations in our method. Our model is non-physically based, thus to specify the physical properties of hair strands such as elastic properties is impractical. We used trial and error to get the desired result. We do not consider the torsion of hair in our model. Torsion has less significant effects for straight hair in common animations, because the external forces that twist straight hair are rare. However, it does occur in curly hair. We would like to improve our model to capture the torsion of hair. One idea is the use of the torsion spring presented in a rigid multibody serial chain [HMT01] which represents a hair strand as a chain of rigid segments could be adapted. Because the hair strand in our model is also represented by a chain of particles, torsion springs between particles could be added to capture the torsion effect. In addition, collisions between hair segments cannot be detected in our model due to the use of sphere-sphere collision detection. We would like to employ a fast and robust collision detection that could find the intersections between lines as well. The stiction force influenced by the electrostatic pull-in also has to be taken into account to generate more realistic results.

Although our method simulates each strand of hair individually, an interpolation technique such as [CJY02] and [BAC*06] can be applied for an application which prefers a better performance to finer detail. For example, in the case of one thousand hair strands with 30k particles (Figure 5), our model can run at 61 fps, which is sufficiently fast for interactive applications such as games.

References

[AUK92] Anjyo K., Usami Y., Kurihara T.: A simple method for extracting the natural beauty of hair. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), pp. 111–120.

[BAC*06] Bertails F., Audoly B., Cani M.-P., Querleux B., Leroy F., Lévêque J.-L.: Super-helices for predicting the dynamics of natural hair. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (2006), pp. 1180–1187.

[BFA02] Bridson R., Fedkiw R., Anderson J.: Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 594–603.

[BNC03] Bando Y., Nishita T., Chen B.-Y.: Animating hair with loosely connected particles. *Comput. Graph. Forum 22*, 3 (2003), 411–418.

[BS05] BOUBEKEUR T., SCHLICK C.: Generic mesh refinement on GPU. In *ACM SIGGRAPH/Eurographics Graphics Hardware* (2005), pp. 99–104.

[CCK05] Choe B., Choi M. G., Ko H.-S.: Simulating complex hair with robust collision handling. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 153–160.

[CJY02] Chang J. T., Jin J., Yu Y.: A practical model for hair mutual interactions. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 73–80.

[DTKT93] Daldegan A., Thalmann N. M., Kurihara T., Thalmann D.: An integrated system for modeling, animating and rendering hair. In *Computer Graphics Forum* (1993), pp. 211–221.

[Har07] Harada T.: Real-time rigid body simulation on GPUs. *GPU Gems 3, Chapter 29* (2007), 123–148.

[HMT01] Hadap S., Magnenat-Thalmann N.: Modeling dynamic hair as a continuum. *Comput. Graph. Forum 20*, 3 (2001), 329–338.

[MHHR07] Müller M., Heidelberger B., Hennix M., Ratcliff J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (2007), 109–118.

[MHTG05] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (2005), pp. 471–478.

[MSW*09] McAdams A., Selle A., Ward K., Sifakis E., Teran J.: Detail preserving continuum simulation of straight hair. In *SIGGRAPH '09: ACM SIGGRAPH 2009 Papers* (2009), pp. 1–6.

[OKN09] Ohta M., Kanamori Y., Nishita T.: Deformation and fracturing using adaptive shape matching with stiffness adjustment. *Comput. Animat. Virtual Worlds 20*, 2-3 (2009), 365–373.

[Osh07] Oshita M.: Real-time hair simulation on GPU with a dynamic wisp model. *Comput. Animat. Virtual Worlds 18*, 4-5 (2007), 583–593.

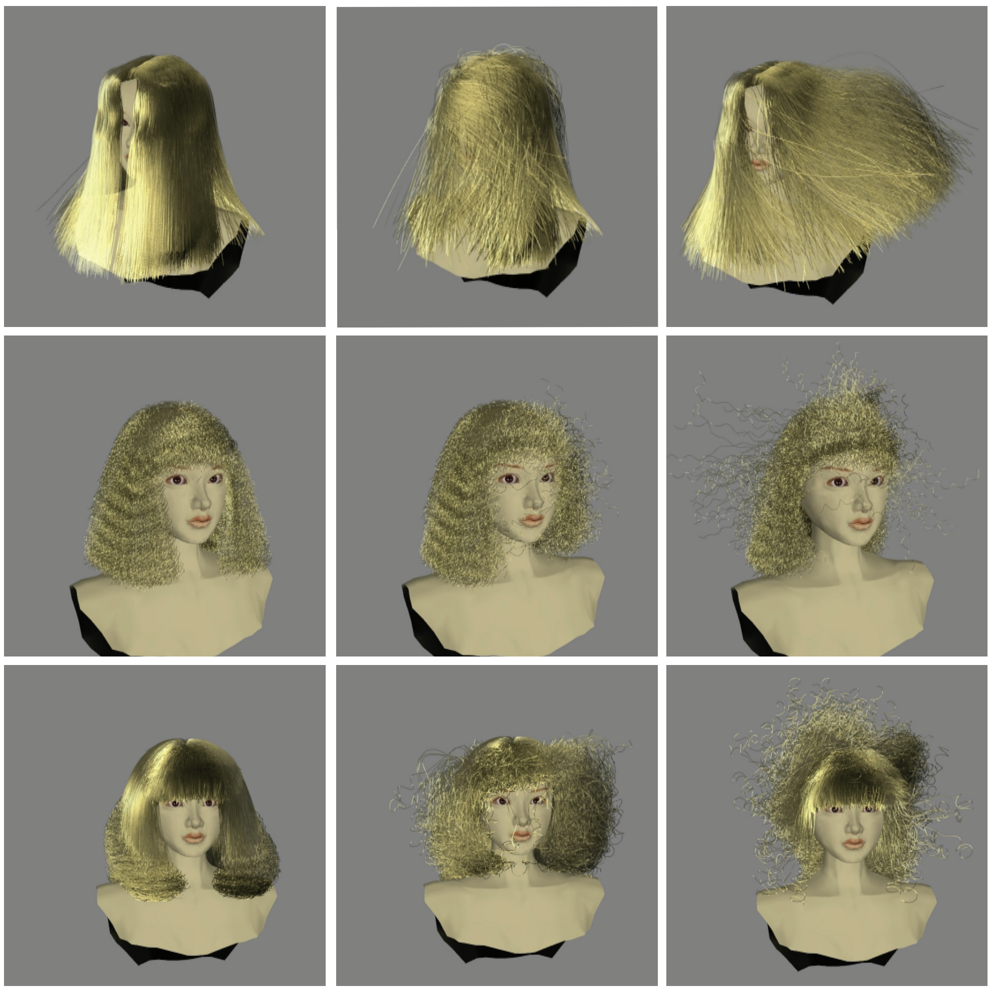

Figure 13: *Animation sequences of straight (top row, 12 fps), curly (middle row, 7 fps) and complex hairstyles (bottom row, 4 fps) in the wind. There are 10k strands (160k particles), 10k strands (580k particles) and 23k strands (764k particles), respectively. The stiffness configurations of the straight and curly hair are $w = 2$ and $w = 5$. Each strand of the complex hairstyle is straight near the roots and curly near the tips with chain region half-widths of $w = 2$ and $w = 6$, respectively.*

[PCP02] Plante E., Cani M.-P., Poulin P.: Capturing the complexity of hair motion. *Graphical Models (GMOD) 64*, 1 (january 2002), 40–58.

[Pro95] Provot X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95* (1995), Davis W. A., Prusinkiewicz P., (Eds.), Canadian Human-Computer Communications Society, pp. 147–154.

[RCT91] Rosenblum R. E., Carlson W. E., Tripp E.: Simulating the structure and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation 2*, 4 (1991), 141–148.

[RJ07] Rivers A. R., James D. L.: FastLSM: fast lattice shape matching for robust real-time deformation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), p. 82.

[SLF08] Selle A., Lentine M., Fedkiw R.: A mass spring model for hair simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (2008), pp. 1–11.

[TB08] Tariq S., Bavoil L.: Real time hair simulation and rendering on the GPU. In *SIGGRAPH '08: ACM SIGGRAPH 2008 talks* (2008), pp. 1–1.

[VMT06] Volino P., Magnenat-Thalmann N.: Real-time animation of complex hairstyles. *IEEE Transactions on Visualization and Computer Graphics 12*, 2 (2006), 131–142.

[WBK*07] Ward K., Bertails F., Kim T.-Y., Marschner S. R., Cani M.-P., Lin M.: A survey on hair modeling: Styling, simulation, and rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 13*, 2 (2007), 213–34.

[YK08] Yuksel C., Keyser J.: Deep opacity maps. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008) 27*, 2 (2008), 675–680.