Hair Simulation Based on Mass Spring System

Yao Lyu

Master Thesis

MSc Computer Animation and Visual Effects

Bournemouth University

NCCA

August 2016

Abstract

In this paper, we present a method for stably simulating stylized hair that addresses artistic needs and performance demands. The paper makes an overview of the mathematical and physical knowledge of basic mass-spring model as well as the techniques to eliminate artefacts. This paper also addresses performance concerns associated with handling hair-hair contact and hair-object collision. Several scenarios are implemented in Houdini using VEX language with the method we described. Finally, the implementation is proven to be robust and suitable for simulating hairs with both regular and irregular objects.

Key words: hair simulation, mass-spring model

Content

1. Introduction

Creating virtual hair has been a prominent topic within the realm of computer graphics. Hair simulation usually includes three parts: hair styling, hair dynamics and rendering. Especially, the hair dynamic has important applications for visual effects, animated features, virtual hair styling and online stores. However, hair dynamic is one of the most challenging phenomena to simulate because of the sheer number of hairs on the head and the complex motion of different hair styles.

Nowadays, researchers focus on the two parts of hair dynamics: single hair dynamic and collective hair dynamic. Single hair dynamic is based on details and intricate parts of hair. But simulating hair separately is complex and tedious. Some authors suggest researching the collective hair behaviour for high efficiency of simulation and rendering. They use clumped models and aggregate hair simulation techniques. These algorithms also have drawbacks. Simulating hairs with irregular geometries can produce hair-hair or hair-body interpenetration sometimes. Additionally, subtle details such as stray hairs, clump separation are not handled by aggregate models. Different algorithms are designed for keeping the specific visual look of the dynamic hair. For example, the helical shape hair, which is a popular feature in animation, is a challenge for developers because it would naturally straighten under stretching. To address the above drawbacks and requirements, we propose using a non-clumped simulation technique in this paper. Though this requires more computation, this allows us to achieve intricate hair behaviour.

This paper attempts to implement one hair solver which could give dependable results with as many individual hairs as possible by using a constitutive model. To do this, this paper uses mass-spring system with point representation which is computationally inexpensive and adept at detecting object collisions. Unfortunately, mass-spring models have difficulty modelling twist. This paper introduces another two springs to keep the curliness of hair. This paper also provides strain limiting approach for avoiding severe deformation and smooth function for stabilizing the hair shape. To handle the interaction of hair and object, this paper uses volume data to detect interpenetration and create friction spring to imitate fiction. This hair simulation system is demonstrated in several examples in Houdini using VEX language.

2. Related Work

Dynamic traditionally involves in working with forces. When it comes to hair, the forces are usually defined as two classes: internal forces which work inside the hair curve and the external forces to represent the interactions with object in the environment. Many previous works have applied mass-spring system to calculate the internal forces with point presentation.

The first approach was came up by [Rosenblum et al. 1991], which used a linear spring for stretching and an angular spring between hair segemtns for bending. [Petrovic et al. 2005] simulated mass-spring hairs as guide hairs, rasterising them onto a level set grid to model interaction and volume, producing uniform hair. [Gupta et al. 2006] used a simplified mass-spring model with lattice point presentation to define a deformation field for embedded rendering hairs. These two algorithms are efficient at modelling bulk behaviour, but they have difficulty modelling discontinuous effects like hair separation, and the highly intricate interacting geometry like curly hair, or individual hair twist. [Selle et al. 2008] introduced a mass-spring model for simulating individual hair as much as possible. He used stretch, bend, twist and altitude springs to form an implied tetrahedron between points, through keeping no-zero altitude spring to prevent volume collapse.

A variety of methods also have been proposed to handle the external forces, like collision impulsion and fiction. [Choe et al. 2005] combined basic mass-spring model with rigid multi-body chain structure to model wisps, and detected collisions by testing the cylinders' volume. Mass-spring models have also been improved with a lattice for hair styling by [gupta et al.2006]. Later the model was modified with an Eulerian fluid solver to keep hair volume and provide a better initial position for the point contacts by [mcadams et al.2009]. Recently hair-body and hair-hair contacts have been more accurately implemented by using a non-smooth Newton solver for Coulomb friction law by [bertails-descoubes et al.2011]. [Daviet et al. 2011] used an analytic solver to solve for Coulomb friction with a hybrid Gauss-Seidel algorithm to ensure convergence.

Similar to prior work, this paper combines [Selle et al. 2008] and [Hayley Iben et al.2013] to build a basic mass-spring model with three springs, including a linear spring for stretch and two springs to keep the curliness and twist. We add perturbed points and construct new springs with additional points to keep the stability of hair structure. For hair interaction, we chose to detect contacts by representing object collision geometry as volume data, which is good at handling irregular geometry efficiently.

3. Constitutive Model

To create the desired hair style, especially the helical shape, this model needs explain two properties of hair at first:

1. Curly hair is similar to spring so the simulation should meet some properties of springs, like it should keep initial curly shape during simulation. While straight hair is similar to string, it should be flexed with motion and wind.
2. Hair could have extension but have maximum length limitation.

To meet the requirements above, this paper introduces three kinds of springs: stretch spring, bending spring and torsion spring. This spring model can work for both straight and curly hair.

3.1 Stretch Spring

We define our hair model using a set of points which are connected by edge springs. These edge springs represent the structure of hair. Let each hair consists of a set of current point positions P = {p_0, . . . , p_{N-1} } and a set of point velocities V = {v_0, . . . , v_{N-1}}.The segment connecting every two points is $e_i = p_{i+1} - p_i$ and the length is *l*. Initial length of each segment at rest pose positions is stored *len0*. We compute a standard damped linear spring force on the edge e_i, as shown in Eq.1.

$$\mathrm{F} = \mathrm{K}_s * (\mathrm{l} - \mathrm{len0}) * \mathrm{normalize}(\mathrm{e}_i) + \mathrm{C}_s * (\mathrm{v}_{i+1} - \mathrm{v}_i) * \mathrm{normalize}(\mathrm{e}_i) \qquad (1)$$

K_s is the spring coefficient and C_s is the damping coefficient. Because springs in our model are connecting two points, each of our spring forces is applied equally to both points in opposite directions on same segment.

3.2 Bending and Torsion spring

Stretch springs could transfer forces from one endpoint to another endpoint through points' movement. However, once force is applied to drag the end of hair, the stretch spring can't help hold the original shape of hair so some segments will be dragged to be straight. Especially the curly hair will be straight in the simulation process. Finally, the hair keeps straight whatever its initial shape is because the stretch forces on every point could keep balanced in the direction of this external force. Therefore, this paper introduces bending spring to stably control the bend between every two points.

During initialization, we construct bending spring between every other point which means every point has at least one bending spring to constrain its curliness. The initial length of each bending spring is recorded at rest pose. The force on every bending is calculated similar to Eq.2.

$$\mathrm{F} = \mathrm{K}_b * (\mathrm{l} - \mathrm{len0}) * \mathrm{normalize}(\mathrm{e}_i) + \mathrm{C}_b * (\mathrm{v}_{i+1} - \mathrm{v}_i) * \mathrm{normalize}(\mathrm{e}_i) \qquad (2)$$

K_b is the bending spring coefficient and C_b is the damping coefficient. Usually, these two coefficients are smaller than according coefficients of stretch spring. Using only stretch and bending spring are insufficient for actual animation when simulating curly hair with a variety of motions. We add the torsion spring to our hair model which connects each point to a point three points away from it so that twist can be modelled. The calculation of torsion spring force is similar to bending spring, while it uses different spring and damping coefficients, Kt and Ct.

These three springs works well on constraining the curliness of every point on hair. The whole spring construction is shown in Fig 3.1.

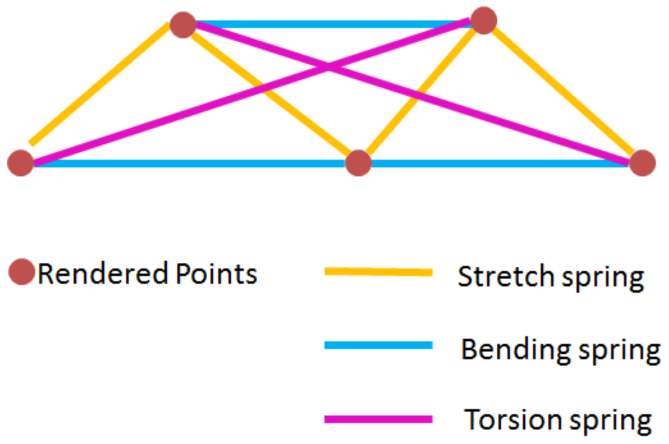

Fig 3.1 Mass-Spring Model

3.3 Smooth function

During the motion of hair, especially curly hair, the helical frequency will be increased like additional rotation of hair is introduced. [Hayley al et.2013] called this artefact as walk cycle, as shown in Fig 3.2. This rotation is physically accurate but presents as artefact in our visual effect which is not desired. Hayley introduces a method to stably generate the material frame by parallel transporting the root frame of the hair along a smoothed piecewise linear curve.

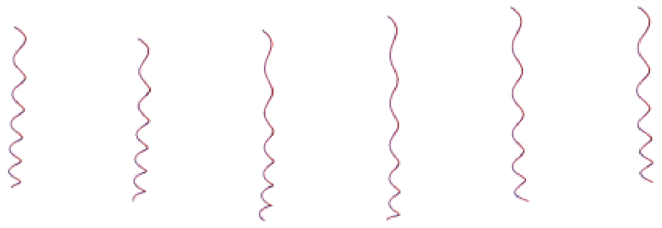

Fig 3.2 This elastic rod model introduces rotation around the helix, apparent in the orientation of the end of the curl.

This smoothing function is defined with an infinite impulse response (IIR) Bessel filter which can produce new result by combining the input and prior values using recursive functions. In this case, we use a set of point position as the input of filter, as shown in Eq.3. Let $\beta = \min(1, \exp(len0/\alpha))$ where len0 is the average rest length per segment of the hair being smoothed, which is stored before dynamic starts. d_i is the vector to modify point *i* for smoothing function. And it is obtained by recursively computing from the points closer to the root to the endpoint.

$$\mathrm{d}_i = 2(1-\beta)\mathrm{d}_{i-1} - (1-\beta)^2\mathrm{d}_{i-2} + \beta^2(p_{i+1} - p_i) \qquad (3)$$

$$\mathrm{p}_i' = \mathrm{p}_{i-1}' + \mathrm{d}_{i-1} \qquad (4)$$

For each point, the first step is to calculate the transformation vector d_i in Eq.3. Usually, $\mathrm{d}_{-2} = \mathrm{d}_{-1} = \mathrm{p}_1 - \mathrm{p}_0$, which means $\mathrm{d}_0 = \mathrm{p}_1 - \mathrm{p}_0$ at i = 0. Subsequent d_i is weighted towards this initial direction. The second step is to modify the position, as shown in Eq.4. p_i' is obtained by last updated points

p_{i-1}' and its transformation vector d_{i-1}. p_0 is the root of the hair curve which is usually glued to the head scalp. Finally, the hair curve is smooth by recursively updating points' position from the locked root position.

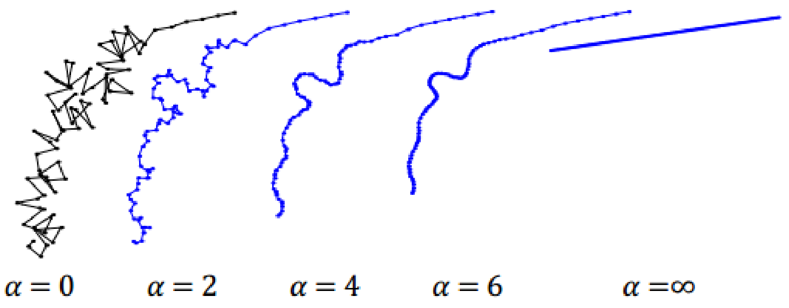

Fig 3.3 Example of a stylized curly hair (far left) and the smoothed curves (blue) computed with α at 2, 4, 6, and ∞ [Hayley al et.2013].

As we can see, the output behaviour is relative to input parameter α. If α= 0, β is set to 1 manually, meaning that $d_i = d_{i+1}$ and no smoothing effect occurs. When $\alpha \to \infty$, then $\beta \to \infty$ and $d_i = d_{i-1} = \cdots = p_1 - p_0$.The effect of smooth function is straight and in the direction of the top segment, regardless of how kinked the input curve is. [Hayley al et.2013] shows how α affects the smooth function in Fig 3.3.

3.4 Improved Mass-Spring Model

Another artefact in hair dynamic is when one endpoint of hair is dragged by the huge force, the hair will go straight, which means that the bending and torsion spring will overlap stretch spring, becoming useless for holding the helical shape. At this situation, additional spring need to be added with points outside the hair curves, which avoids the same problem met by bending and torsion springs.

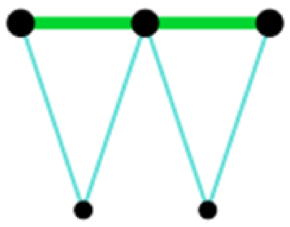

Fig 3.4 Perturb points(below) from two neighbour segments in improved mass-spring system.

[Andrew al et.2008] proposed to perturb middle points for two con-linear hair segments. He creates two new points at the midpoints of the two neighbour segments and perturbs them to form two triangles as shown in Fig 3.4. These triangles should be fairly rigid so they are given springs on their edges that are as strong as the hair stretch springs. From these triangles we can add updated bending and torsion springs to form a full hair model as shown in Fig 3.5. Extra stretch spring and updated torsion spring with perturbed points ensure that there are at least some stretch springs and torsion springs are not parallel to force direction during simulation. Thus instead of using an explicit coordinate system we model an implicit one by using offset points together with extra springs, causing a marginally higher simulation cost, but still fitting into a simple mass-spring framework.

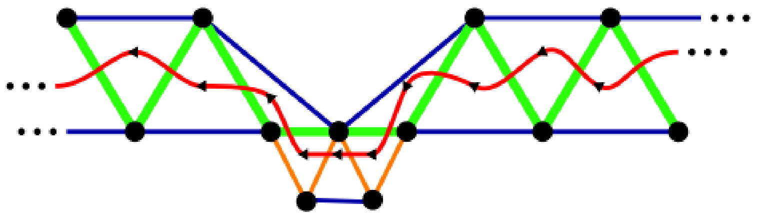

Fig 3.5 Improved mass-spring system with perturbed points

Thus our algorithm prepares a hair curve for simulation by first sampling discrete points p_0 , . . ., p_i equally in arclength. We perturb new points off of the original curve such that the edge lengths of the newly created triangles are equal to the length of the parent segment. In addition, if there are many consecutive segments with perturbed particles, we rotate the perturbed points about the hair axis to ensure good direction sampling. The improved spring structure is shown in Fig 3.6.

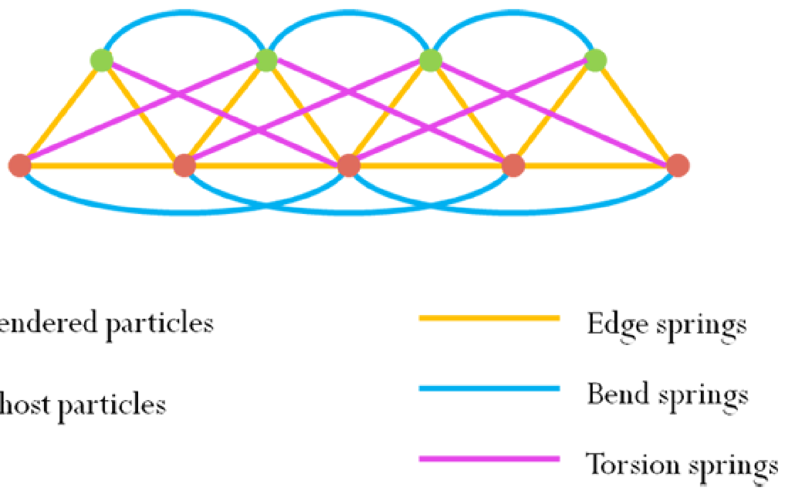

Fig 3.6 Improved mass-spring structure

4. Time integration

Given our hair model, time integration from time t_n to t_{n+1} proceeds as follows:

- Step1: $v_n=v_{n-1}+a*\Delta t$
- Step2: modify vn with strain limiting
- Step3: $x_n=x_{n-1}+v*\Delta t$
- Step4: Body collision modify v_n and xn
- Step5: Self-collision modify vn and xn

The acceleration in Step1 is calculated using explicit time integration after spring model. The velocity computed in step 1 is processed with strain limiting method(in Chapter 5) before being used to predict position. After position update, collisions are applied which discussed in Chapter 6. The velocity before step 4 is discarded and is evolved in last two steps.

5. Strain limiting

Complex head motions can cause severe stretching especially in springs which have one of their two endpoints constrained to a character’s head. [Caramana et al.1998] said that a triangle edge should not change length by more than 10% in a single time. Hayley introduced the core spring that controls the longitudinal stretch of the curls without stiffening the bending springs. It works as normal spring but the spring coefficient is set from 0 to full value to avoid adding unnecessary constraints. [Bridson et al. 2002] used strain limiting approaches to alleviate high strain in cloth, that apply momentum conserving velocity impulses to points attached by springs that exceed 10% deformation. This can be enforced by either adaptively decreasing the time step or decreasing the strain rate.

In this paper, we correct the position and velocity to protect the spring under large acceleration. Since correcting one spring potentially damages another, iteration is typically used. In our case, hair is relatively light compared to the head so that the acceleration usually comes from the head’s movement. Therefore, a biased strain limiting approach usually is employed that marches from the root of the hair and projects the length by moving only the point further from the root in the direction formed by the two point.

During implementation, we obtain the prediction of current position and velocity since the internal forces are already knew. Then we handle the strain limiting operation based on the predicted position. In any step, the maximum length of two points is 1.1 times initial spring length. If the predicted length of two points is greater than the maximum, we project the points backward. Then we update the velocity on the basis of last position and modified position. Note that the strain limiting in step 2 above only affects the velocity used to update the positions and has no direct effect on the velocity used for evolution in later steps.

6. Collision Detection

This chapter deals with the difficult problems of hair interactions, which play a major role in the motion of a full head of hair. Hair interaction includes hair-hair interaction and hair-object collision.

6.1 Hair-hair interaction

Considered the special property of hair, hair-hair interaction is not the simple repulsion of collision, which also includes friction and static charge adhesion which cause hair to stick together.

To detect hair collision and contact, the distance of each segment on neighbour hair is compared. If consecutive points x1 and x2 lie on hair strand h1, and consecutive points x3 and x4 lie on hair strand h2, the smallest distance between the hair segments x21 and x43 (where x21 = x2 - x1, x43 = x4 - x4) need to be calculated. This distance is the line mutually orthogonal to the two segments. If the endpoints of this line are p1 and p2 on h1 and h2 respectively, the equations are shown below:

$$x_{21} = x_2 - x_1 \tag{5}$$

$$x_{43} = x_4 - x_3 \tag{6}$$

$$p_1 = x_1 + ax_{21} \tag{7}$$

$$p_2 = x_3 + ax_{43} \tag{8}$$

$$0 \leq a, b \leq 1$$

Where a and b refer to the relative positions of p1 and p2 on h1 and h2 respectively.

If the distance between p1 and p2 is within the collision distance, the impulsion is applied to the relative segments on h1 and h2, calculated as below.

$$\tilde{I} = \frac{2I}{a^2+(1-a^2)+b^2+(1-b^2)} \tag{9}$$

$$v_1 = v_1 + (1-a)(\frac{\tilde{I}}{m})normal(n) \tag{10}$$

$$v_2 = v_2 + a(\frac{\tilde{I}}{m})normal(n) \tag{11}$$

$$v_3 = v_3 - (1-b)(\frac{\tilde{I}}{m})normal(n) \tag{12}$$

$$v_4 = v_4 - b(\frac{\tilde{I}}{m})normal(n) \tag{13}$$

If the distance between p1 and p2 is within the contact distance, then a stiction spring is created between the two points. [Jimenez and Luciani 1993] said that springs can be useful for making and breaking dynamic constraints. Therefore this spring works like normal sprins this paper described in Chapter 3 so that it can be stretched and compressed until the length of the spring exceeds a separation threshold. Connections will be broken as hairs separate beyond the distance threshold.

Handling stiction spring is quiet intractable. Firstly, the two endpoints of stiction spring should be added on the segments so that they change the structure of the hair strand and affect the structural stretch springs. The whole stretch spring model should be modified. Secondly, the computation for all stiction springs can get really complicated and cause the hair to blow up if there are as well as too many stiction springs on the whole hair strand. Thirdly, when the connection between two edges break, it is also necessary to remove the stiction spring. This requires removing the temporary points we created, removing the edge springs connecting those temporary points, and adding back the old stretch springs. Taken the situation above into consideration, this project doesn’t implement the stiction spring when one hair contacts with another hair.

6.2 Hair-Object Collision

The collision objects are usually head or body. To detect the collision, the geometry is usually represented as a level set signed distance function which aims to discriminate whether one point is inside the geometry. Most projects use a simple sphere to represent the head, which means only measuring the distance from any position to sphere centre could make sure whether this point goes inside the geometry. Using sphere is simple for collision detection but raises the problem of irregular geometries. This paper is looking for one efficient method to represent the irregular collision geometry as distance data, especially during the motion.

Openvdb provides a hierarchical data structure to describe sparse, time-varying, volumetric data discretized on three-dimensional grids. It could generate volume data from geometry robustly and effectively. The volume density differs from the distance between points and the centre of geometry, which is 1 on the geometry centre, 0 on the geometry skin and 0 outside the skin.

We adapt the techniques from the volume density through Openvdb library. At Step 4 in time integration, we obtain the density value of position x_n. If the density value is greater than 0, this point is going to collide the geometry, otherwise it will to stay outside the geometry.

As mentioned hair is a soft and light material compared to body and head, the impulse affects hair’s velocity which will be set to 0 when collision happens, however, it doesn’t affect the movement of head or body. At the same time, due to contact between hair and collision object, hair experience static friction, so a friction force is need to be applied. Similar to hair-hair stiction spring, the stiction spring could be created to imitate the friction effects. If the point has already penetrated the object, stiction spring applies a penalty force to push the point out of the object. Otherwise, after collision when point still stays within the small distance, the stiction works by using this spring to drag the point back to skin. This force is calculated as below:

$$\mathrm{F}_r = -1 * K_s * (p_i - o_j) * normal(p_i - o_j) \tag{14}$$

F_r is the friction spring force. K_s is the spring constant of the penalty spring, p_i is the point going to collide the object and o_j is the point on object skin collided by p_i. The normal represents the direction of the contact.

7. Implementation

This chapter covers how the techniques explained in the previous are implemented. The implementation of this project is done in Houdini based mainly on the use of DOPs, and VEX language. The main task of implementation is to add one hair solver in Houdini then use simple hair model to testify the stability and efficiency of the solver.

Modellin
Hair Dynamic
Rendering

Fig 7.1 Implementation steps

We divide the whole implementation into three steps, as shown above in Fig 7.1:

In the first step, we create at least two geometries: hair and object for collision. The hair is attached to the chosen scalp of the object.

In the second step, one DOP network is implemented inside the system of hair simulation which calculates the internal and external forces for hair. The collision and strain limiting will be detected in each time step.

Finally, in line with our goal to simulate the full head of hair, all hairs are visualised and rendered inside Houdini without new hair generation.

7.1 Modelling

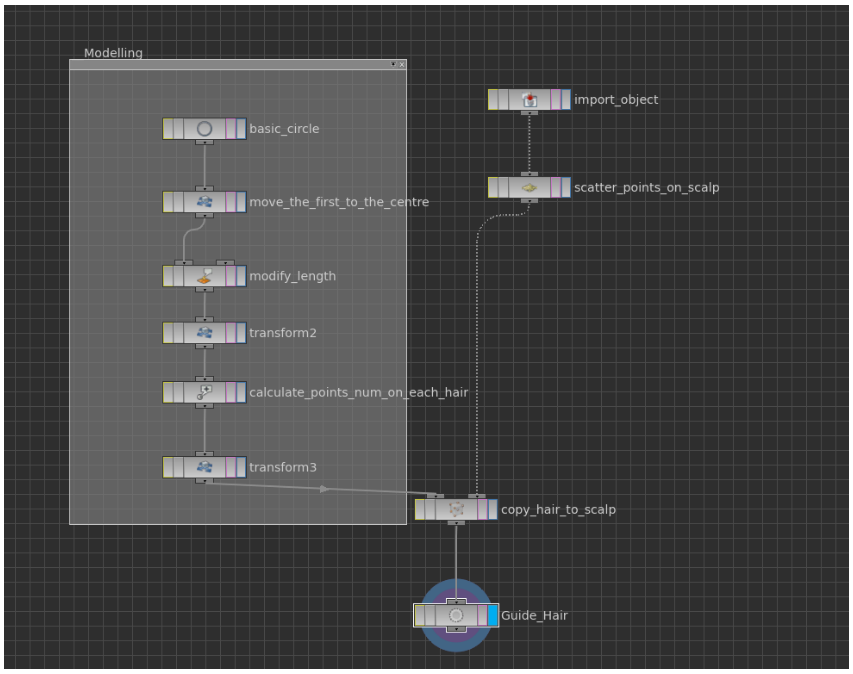

Fig 7.2 Modelling Houdini network

As shown in Fig 7.2, we start from modelling single curly hair by using circle node(basic_circle) for the basic circles and point node(modify_length) to stretch circles in y axis. In order to process the spring model and collision operation, the single hair psolyline is divided into segments with point presentation. More points are, more complex the dynamics will be. In this project, we set the hair segments with same length. The segment length and total points' number of single hair are recorded at rest position.

For the improved spring model proposed in Chapter 3, additional points need to be created around the original hair. We perturb one point for each hair segment which could form an equilateral triangle with two hair points. In addition, we rotate the perturbed points about the hair axis to ensure good direction sampling. Fig7.3 shows the structure with perturbed points.

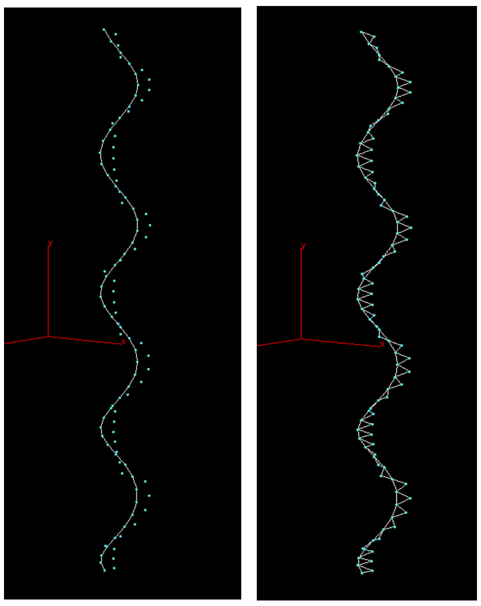
Fig 7.3 Left pic shows the perturbed points. Right pic shows the additional stretch springs constructed with perturbed points.

Improved spring structure introduces two more edge springs for each point which could help to hold smoother hair shape in hair dynamics, while it leads to extra computation and difficulty to process the forces. Therefore, we implement the normal spring model in Section 3.1 and Section 3.2 for whole body hair simulation and the improved spring model in Section 3.4 for single hair simulation. The last step of modelling is to glue multiple hairs to scalp inside Houdini hair tool. As shown below, hairs are applied in different collision objects.

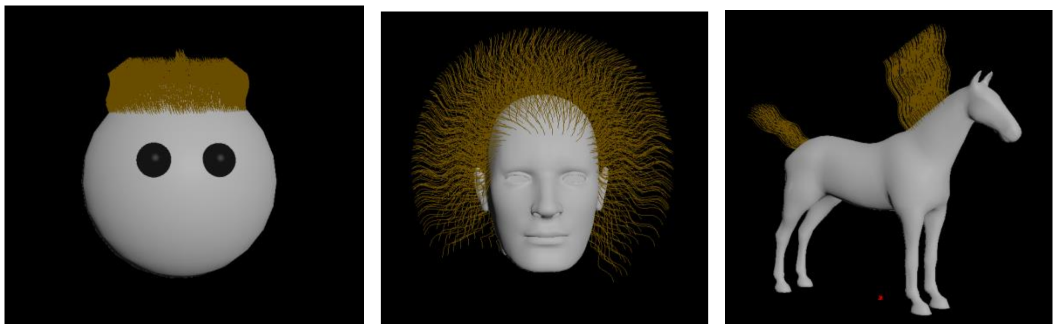
Fig 7.4 Hair models

7.2 Hair Dynamic

Houdini has hair system which doesn't have dynamic solver inside. We implement our algorithms described in section xxx and section in Houdini's hair system. The whole hair system is shown in Fig 7.5. We divide the whole dynamic work into two parts: pre-solver on rest pose and dynamic solver which are implemented in **guides** node and **guide_dynamics** node respectively. This paper aims at simulating a full head of hair so we regard the guide hair in network as our full head hair without rendering new hairs.

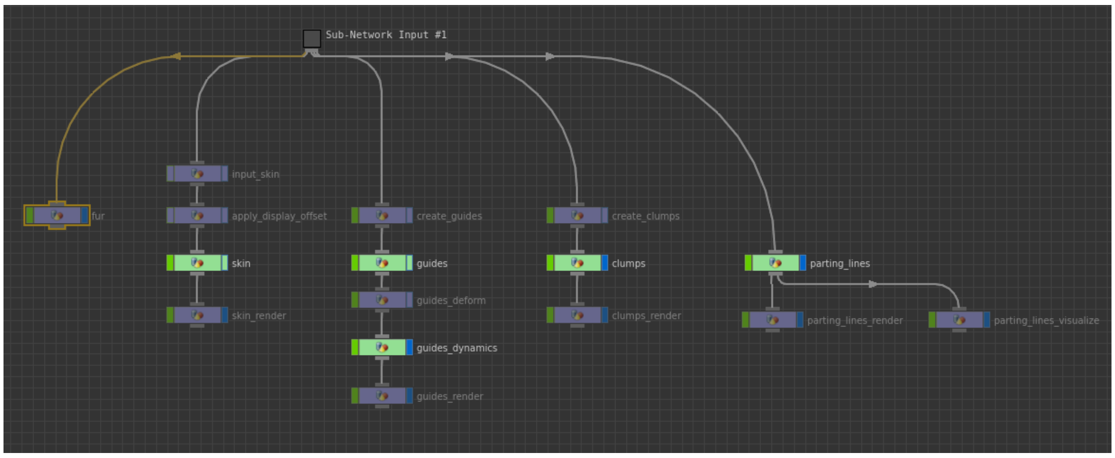

Fig 7.5 Hair Dynamic Network

7.2.1 Hair pre-solver in rest pose

Before the real dynamic starts, several parameters which represent the initial situation of springs need to be calculated, as shown in Fig 7.6. In node **Initialize_length**, three kinds of initial length (for stretch spring, bending spring and torsion spring) are computed for spring model using VEX language. The local number of each point on its hair is recorded by node **Calculate_local_number** for simply building the spring network. Now, the curve is ready to be used for the calculation of the dynamics of the hair.

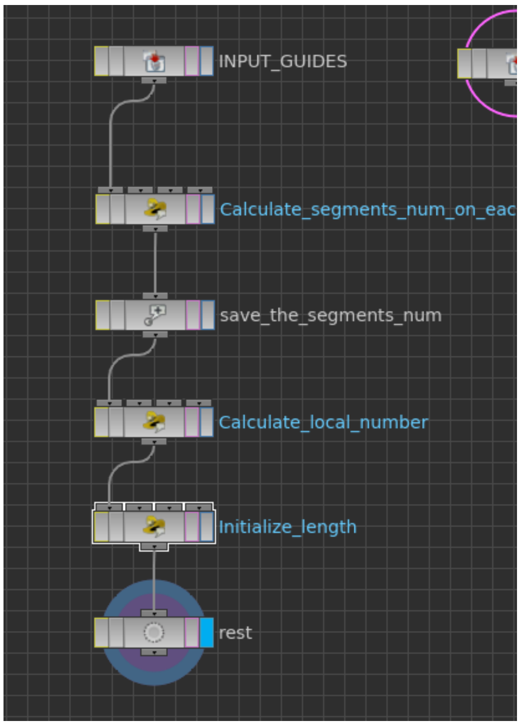

Fig 7.6 Hair pre-solver in rest pose

7.2.2 Hair dynamic solver

All dynamic steps are implemented inside **guide_dynamics** node with one DOP network, as shown in Fig 7.7. The DOP network consists of two parts: internal forces calculation and external forces calculation. Fig 7.8 shows the DOP network. All forces are applied to points which are not glued on the scalp. We follow the time integration explained in Chapter4.

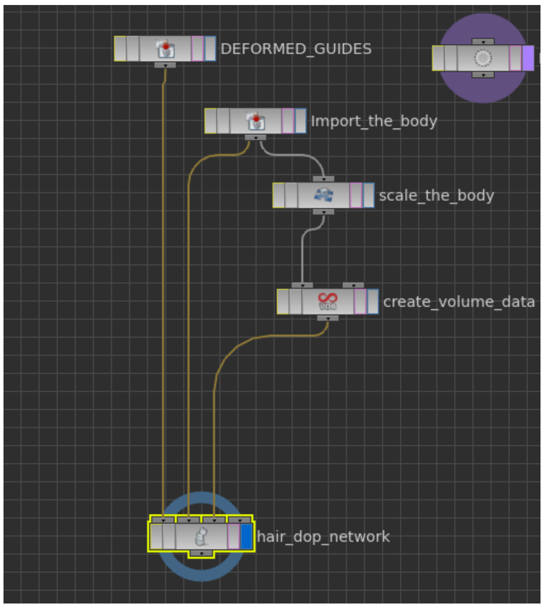

Fig 7.7 Guide_dynamics node network

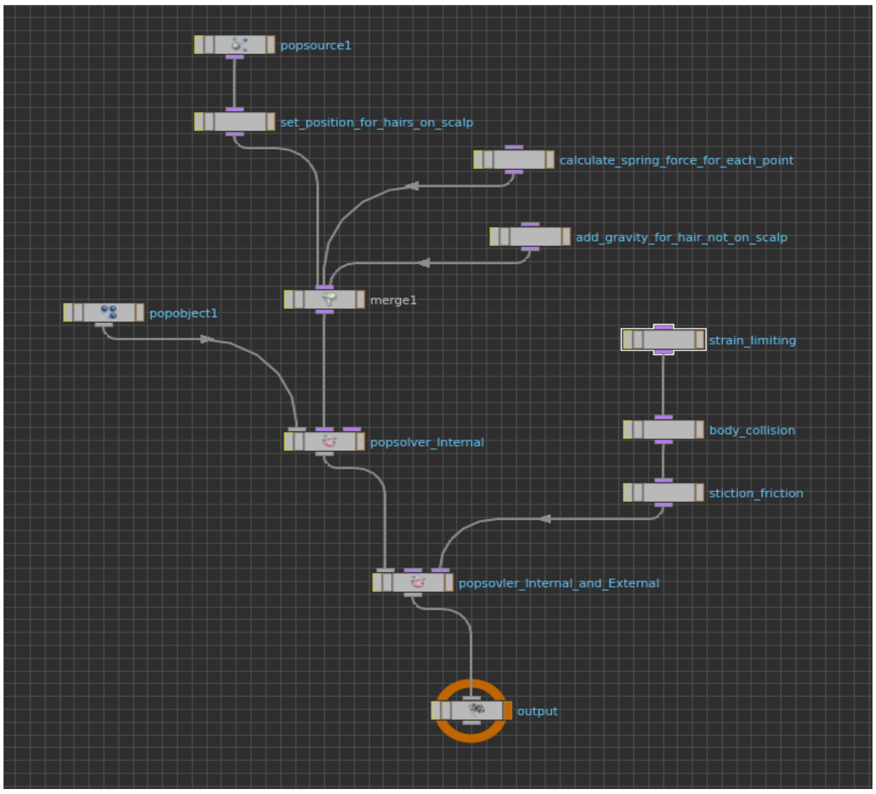
popsource1
set_position_for_hairs_on_scalp
calculate_spring_force_for_each_point
add_gravity_for_hair_not_on_scalp
merge1
popobject1
strain_limiting
body_collision
popsolver_Internal
stiction_friction
popsovler_Internal_and_External
output

Fig 7.8 DOP network

Internal forces of each point include three kinds of spring forces and gravity. Spring forces are calculated in **calculate_spring_force_for_each_point** node by mainly using **SpringForce()** function. The calculation process is shown in Fig 7.9. Since stretch spring, bending spring and torsion spring use same equation in spring model, they can be computed in **SpringForce()** at the same time with different spring and damping coefficients. We list these coefficients in the figure. Then add gravity to points in **add_gravity_for_hair_not_on_scalp** node. Finally, use a **popsolver** to integrate internal forces and update the velocity.

```
For each point:
    For each spring it connected:
        get another endpoints' position
        get another endpoints' velocity
        use SpringForce() to calculate spring force
        decide force direction
    Add all spring forces
```

External forces include the force to drag points backward when severe deformation happens in strain limiting and the friction force during collision. In **strain_limiting** node, we check the length of each segment and adjust the position of point which is farther from the root if current length exceeds 10% deformity. Strain limiting operation doesn't introduce new force indeed and uses position update directly. Modifying point's position reduces the unnecessary computation brought by force and velocity integration. Both of them are useless for next step, as explained in Chapter 4.

Collision detection should be handled after strain limiting operation. Any object could be used for collision in the environment due to our effective Openvdb library. First, we generate the volume density for collision object outside DOP network and import this volume data into DOP network. In each time step, we check the corresponding density of each hair point in **body_collision** node. If it is greater than 0, this point is going to collide the object. We set the velocity of this point to zero and create the friction spring which connects the object skin and the hair point in **stiction_friction** node. Finally one **popsolver** is used to merge the external forces and internal forces. Final velocity and position are updated by the popsolver.

8. Results and Discussion

In this chapter we analyse the results generated in this project. We discuss the efficiency, performance and parameters used for several scenarios: single hair, single movement, human head and horse.

The first effect is the single hair. We hang two hairs at the same height and simulate them with normal spring model and improved spring model respectively. We toot screenshot at frame 50 to show the result in Fig 8.1. The left is normal spring model and the right is improved spring model. We could clearly see that normal model can easily get deformed and can’t keep the initial helical shape. The improved model acts more realistic than normal model.

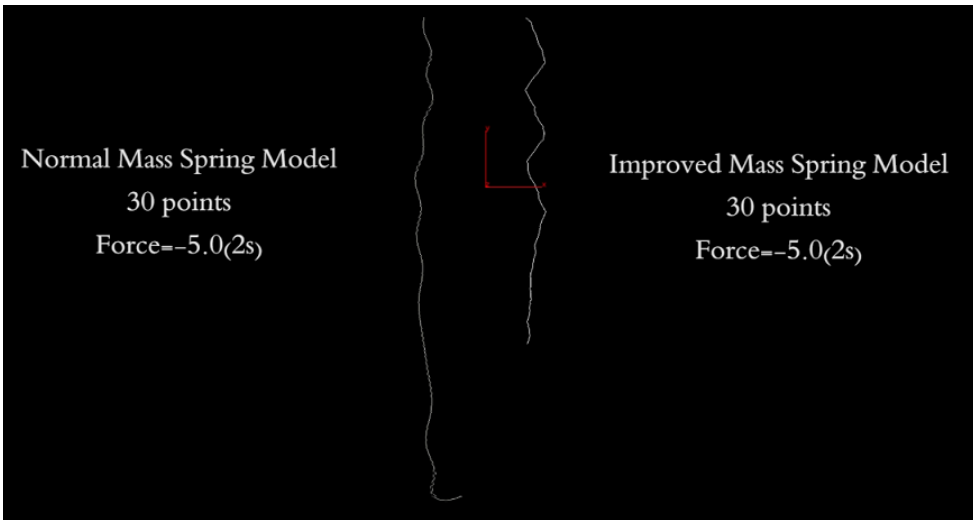

Fig 8.1 Single Hair Simulation

The second effect is simple object movement. We have three different scenarios. Firstly, we simulate a simple sphere with 1000 hairs with 30 points each. As shown in Fig 8.2, hair could handle collision with simple still object. Secondly, we compare two models which one has strain limiting and another doesn’t. We could see strain limiting controls the unlimited hair deformation. Thirdly, we apply transformation and rotation to sphere head. The hairs’ movement and interaction are very precise, even with eyeballs, shown in Fig 8.4.

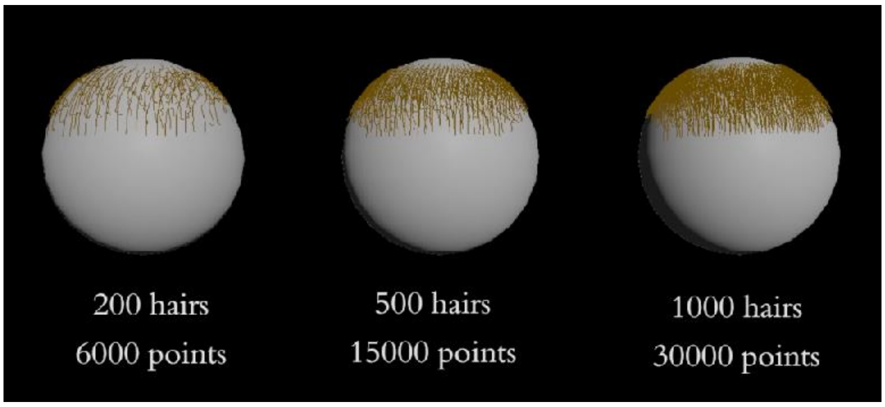

Fig8.2 Collision Detection

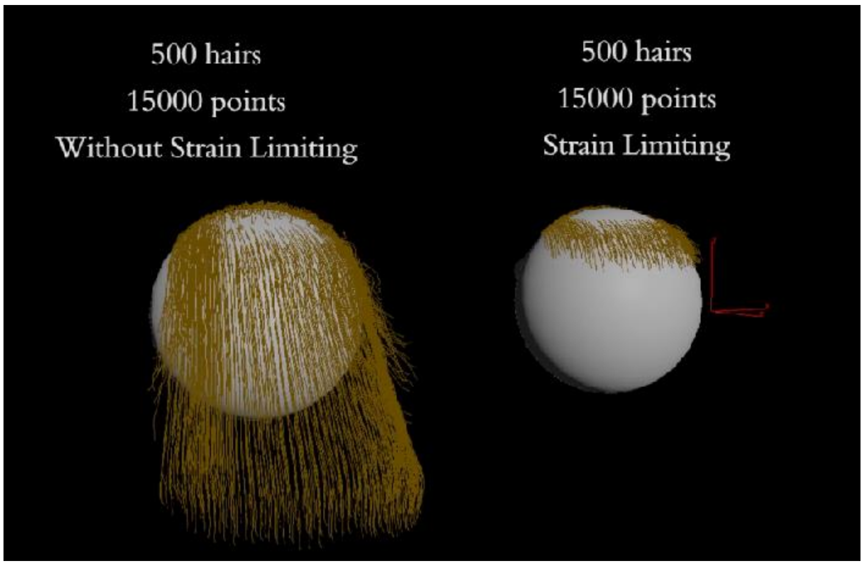

Fig 8.3 Strain Limiting Method

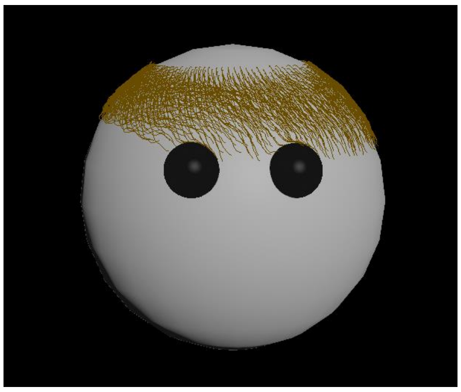

Fig 8.4 Simple Object Movement

The third effect is irregular object movement. We have two different scenarios also. Firstly, we import a male head as collision object to observe the effect when hair falls down, as shown in Fig 8.5. 1500 hairs with 30 points each are simulated. After the collision, hair will stick to the head accurately. However, the artefact happens around the ears sometimes. That is because our voxel size is not small enough which leads to some volume data around object skin is inaccurate. However we can’t obtain efficiency and accuracy at the same time. Therefore, at this case, we neglect the artefact around ears. Finally, we model a horse as collision object and glue hair as its tail and to its back. Hairs with non-uniformed length are applied in this case, which also could be simulated well. The result is shown below.

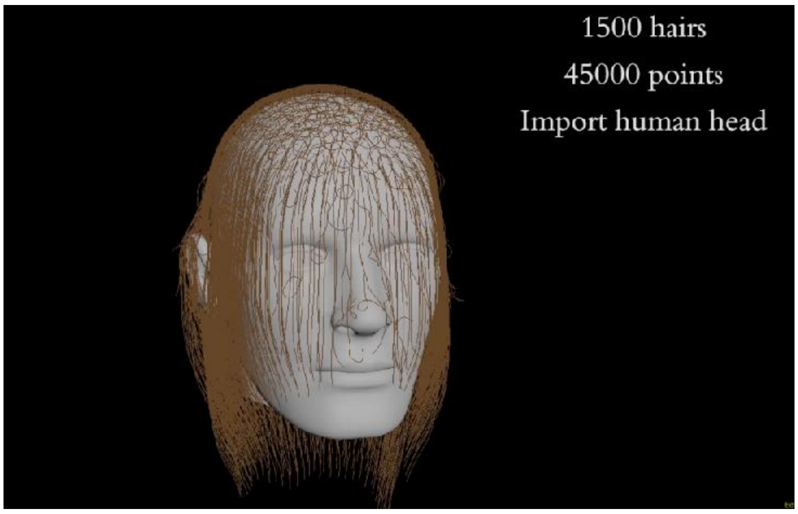

Fig 8.5 Human Head Application

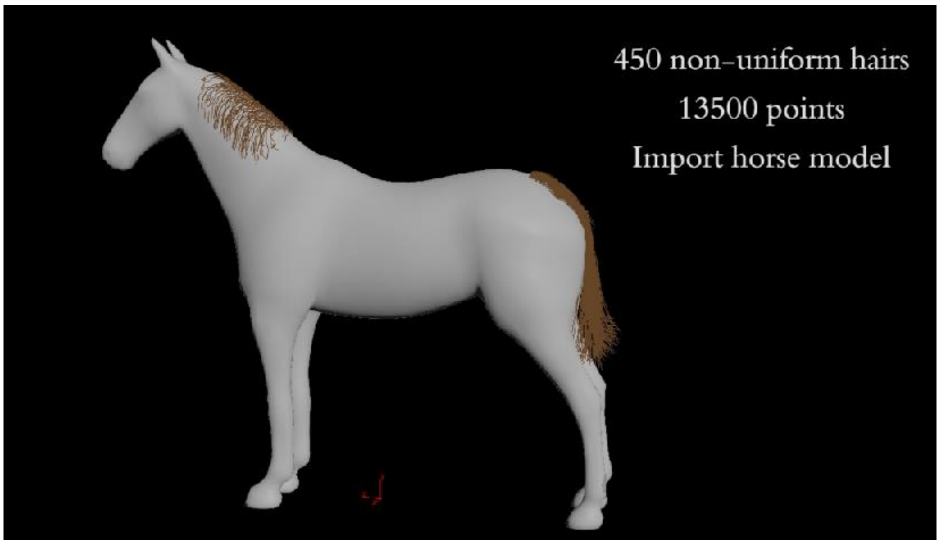

Fig 8.6 Non-uniform Hair Application

9. Conclusion and Future work

This paper mainly implements one hair solver which could give dependable results with as many individual hairs as possible by using the mass spring model. Several techniques are used to eliminate the artefacts in simulation as well. Volume data is introduced in collision detection part. This hair simulation system is demonstrated in several examples in Houdini using VEX language.

At the end of this project, it is meaningful to evaluate this project and analyse what we could do to improve it in future.

Firstly, due to time limitation and complexity, this project doesn't implement the hair-hair interaction described in Chapter 6, which leads to hair volume collapse in our effect, where hair behaves like strings and couldn't keep specific hair style after falling on the scalp. Several methods can be used to fix this problem in future, including the theory we mentioned in Section6.1 and Position Based Dynamics which works on position directly. Secondly, to save simulation time, we set the voxel size as 0.1 to compute volume data which is not accurate enough to represent the boundary of inside and outside geometry, and also easy to create artefacts around small details, like eyes, ears and nose. Thirdly, we should adjust the spring coefficients for different applications. As we can see in the final video, the hair in "human head" scenario is a little soft compared to real hair, while the hair in "horse" scenario is realistic. All these works need to be done in future to improve the performance and efficiency of hair simulation.

Reference

Andrew Selle, Michael Lentine, and Ronald Fedkiw, 2008. A Mass Spring Model for Hair Simulation. ACM Transactions on Graphics SIGGRAPH 2008, ACM TOG, 27, 64.1-64.11.

BERTAILS-DESCOUBES, F., CADOUX, F., DAVIET, G., AND ACARY, V. 2011. A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. ACM Trans. Graph. 30 (February), 6:1–6:14.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. ACM Trans. Graph. 21, 3, 594–603.

CARAMANA, E., BURTON, D., SHASHKOV, M., AND WHALEN, P. 1998. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. Journal of Computational Physics 146, 227–262.

CHOE, B., AND KO, H.-S. 2005. A statistical wisp model and pseudophysical approaches for interactive hairstyle generation. IEEE Trans. on Vis. and Comput. Graph. 11, 2, 160–S170.

DAVIET, G., BERTAILS-DESCOUBES, F., AND BOISSIEUX, L. 2011. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In Proc. of the 2011 SIGGRAPH Asia Conference, 139:1–139:12.

GUPTA, R., MONTAGNOO, M., VOLINO, P., AND MAGNENATTHALMANN, N. 2006. Optimized framework for real time hair simulation. In CGI Proc. 2006, 702–710.

Hayley Iben, Mark Meyer, Lena Petrovic, Olivier Soares, John Anderson, and Andrew Witkin, 2013. Artistic simulation of curly hair. In Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA ’13, 63–71.

JIMENEZ, S., AND LUCIANI, A. 1993. Animation of interacting objects with collisions and prolonged contacts. In Modeling in computer graphics—methods and applications, Springer-Verlag, B. Falcidieno and T. L. Kunii, Eds., Proc. of the IFIP WG 5.10 Working Conf., 129–141.

MCADAMS, A., SELLE, A., WARD, K., SIFAKIS, E., AND TERAN, J. 2009. Detail preserving continuum simulation of straight hair. In ACM SIGGRAPH 2009 Papers, 62:1–62:6.

PETROVIC, L., HENNE, M., AND ANDERSON, J., 2005. Volumetric methods for simulation and rendering of hair.. Tech. Rep., 06-08.

ROSENBLUM, R. E., CARLSON, W. E., 1991. Simulating the structure and dynamics of human hair: modelling, rendering and animation. J. Vis. and Comput. Anim., 2, 4, 141–148.