Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2013)
J. Bender, J. Dequidt, C. Duriez, and G. Zachmann (Editors)

Tridiagonal Matrix Formulation for Inextensible Hair Strand Simulation

Dongsoo Han[1] and Takahiro Harada[1]

[1]Advanced Micro Devices, Inc.

Abstract

This paper proposes a method to simulate inextensible hair strands using tridiagonal matrix formulation in which distance constraints are formulated as a linear system. The proposed method avoids constructing a full matrix explicitly. Instead, it takes advantage of the chain topology and serial indexing to formulate symmetric tridiagonal matrix. Furthermore, we use a linear distance constraint so that the constraint gradient can be easily formulated. With this matrix-free formulation, memory usage can be extremely lowered. Since the formulated matrix is diagonally dominant, we can solve it by an efficient direct solver. Comparing error (i.e., stretch of constraints) of the proposed constraint solver to ones of the position-based solver with different number of iterations, we show that error of the proposed method is much smaller than those of position-based solver. Also the simulation result shows mush less numerical damping compared to Dynamic Follow-The-Leader method. By implementing in GPU, we demonstrate that our proposed method is simple and efficient.

Categories and Subject Descriptors (according to ACM CCS):
I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Parallel

1. Introduction

Thanks to recent advancements in real-time rendering techniques, an experience in a video game is getting more and more realistic. However, it is still challenging to simulate tens of thousands hair or fur in real-time. Especially user-controllable game characters can get extremely large external forces by sudden spinning or jumping and it can cause visually disturbing elongation. Even with many iterations in position-based method, it is hard to enforce distance constraints.

In this paper, we propose a method to simulate an inextensible hair strand using tridiagonal matrix formulation (TMF). The proposed method can enforce almost zero stretch with just two sequential sweeps on constraints, therefore the complexity of the algorithm is low compared to iterating over constraints multiple times using position-based solver [MHHR07]. Also our method shows very low numerical damping which is is a side effect of Dynamic Follow-The-Leader method [MKC12].

In the following sections, we will first formulate a general constraint linear equation. Second, we will look into a special case of hair constraint and simplify the linear equation by carefully choosing a linear constraint and serial indexing. By this, instead building a full matrix explicitly, we can calculate non-zero coefficients directly from connected constraints. Then we will show that a computationally inexpensive direct solver is applicable for the linear system. Although the proposed method formulates a linear system and solves with a direct solver, the complexity of the proposed algorithm is low. Thus, it is easy to implement. This is one of the advantages we can get from the proposed formulation. As a proof of it, we show a GPU implementation of the algorithm.

2. Related Work

Several approaches have been studied to simulate inextensible hair, fur or ropes and stiff spring system was widely used as in [RCT91, SLF08, CCK05].

Baraff and Witkin [BW98] proposed an implicit integration method to use large timesteps with stiff springs for cloth

DOI: 10.2312/PE.vriphys.vriphys13.011-016

simulation. Implicit integration was later used in the case of hair simulation [WL03, CCK05, CCK05]. Even with stiff springs, it often requires a post-correctional process to limit the stretching of springs [Pro96, BFA02].

Another approach is to use constraint mechanics to solve for Lagrange multipliers to enforce inextensibility in the global manner. House et al. [HDB96] used constrained dynamics simulation techniques for cloth. Goldenthal et al. [GHF*07] developed the Fast Projection method for cloth simulation. Spillmann and Harders [SH10] applied the constraint mechanics to rope simulation. Generally this constraint dynamics requires to construct constraint gradient matrix and a linear equation explicitly. To solve the linear equation, usually an expensive pivoting-based direct solver is used such as PARDISO [Sch06] solver.

Müller et al. [MHHR07] introduced a position-based dynamics (PBD) which solves the constraint dynamics problem in an iterative manner. The method is suitable for real-time applications because of its simplicity and stability. However, a drawback of the method is a poor convergence inherited from Gauss-Seidel iterative solver. To improve the convergence, a hierarchical solver is proposed in [Mül08]. This method uses low resolution representations of a cloth to propagate the error correction over the cloth faster. However, it can produce artifacts if low resolution meshes are ill shaped. Müller et al. [MKC12] exploits the chain structure of a hair strand to achieve a fast convergence of position-based solver. However, the method generates an undesirable artificial damping. Kim et al. [KCMF12] uses additional constraints to improve the convergence of position-based solver. Han et al. [HH12] used the position-based solver for inextensible hair in which they introduced local and global shape constraints to simulate styled hairs and help the convergence of distance constraints efficiently.

Our method formulates the hair simulation using constraint mechanics as in [GHF*07]. However, we exploit chain structure and serial indexing to formulate a tridiagonal matrix which can be solved by an efficient direct solver. Out method is closely related to [MKC12] but addresses the artificial damping problem.

Besides inextensibility, hair simulation needs bending and twisting effects to represent various hair styles. Various approaches have been developed such as springs [CCK05, SLF08, IMP*13], constraints [HH12] and shape matching [RKN10]. More information can be found in [WBK*07]. Also hair-hair and hair-object collision handling [CCK05, HMT01, MSW*09] is an important part. Since this paper is an extension of our previous hair research [HH12], we do not cover hair styles and collisions.

3. Method

To simulate an inextensible hair strand, we model it as a hard constraint problem. This section begins with the formulation of general constraint dynamics. Next, we look into chain structure case and tridiagonal matrix formulation. Finally, we show that a computationally inexpensive direct solver can be used to solve the linear system.

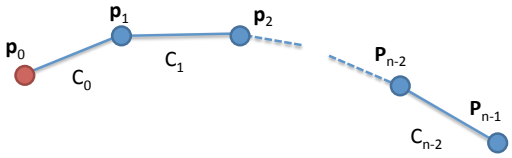


Figure 1: Hair or fur chain structure. The first vertex (red) has infinite mass.

3.1. General Formulation

Constraint $\mathbf{C}$ is a vector containing distance constraints between two connected vertices. $\nabla\mathbf{C}$ is a constraint gradient matrix which has $m \times n$ dimension (m is a number of constraints and n is a number of vertices). By taking up to first order Taylor expansion, $\mathbf{C}(\mathbf{P}+\Delta\mathbf{P})$ can be expressed as $\mathbf{C}(\mathbf{P}+\Delta\mathbf{P}) \approx \mathbf{C}(\mathbf{P}) + \nabla\mathbf{C}(\mathbf{P})\Delta\mathbf{P} = 0$ where $\mathbf{P}$ is a vector storing all the vertex positions. Thus,

$$\nabla\mathbf{C}(\mathbf{P})\Delta\mathbf{P} = -\mathbf{C}(\mathbf{P}) \tag{1}$$

By principal of virtual work of constraint force, position increment $\Delta\mathbf{P}$ can be formulated as

$$\Delta\mathbf{P} = -h^2\mathbf{M}^{-1}\nabla\mathbf{C}(\mathbf{P})^T\Delta\lambda \tag{2}$$

where $\Delta\lambda$ is increment of Lagrange multipliers, h is a timestep and $\mathbf{M}^{-1}$ is an inverse of mass matrix.

From Eqn. 2 and Eqn. 1, we can obtain a linear system and compute $\Delta\mathbf{P}$. $\Delta\lambda'$ is $h^2\Delta\lambda$.

$$\nabla\mathbf{C}(\mathbf{P})\mathbf{M}^{-1}\nabla\mathbf{C}(\mathbf{P})^T\Delta\lambda' = \mathbf{C}(\mathbf{P}). \tag{3}$$

$$\Delta\mathbf{P} = -\mathbf{M}^{-1}\nabla\mathbf{C}(\mathbf{P})^T\Delta\lambda' \tag{4}$$

Naïve implementation of Eqn. 3 is done by building a full matrix, $\nabla\mathbf{C}(\mathbf{P})$ and multiplying it with inverse of mass matrix and transpose of $\nabla\mathbf{C}(\mathbf{P})$. However, we can directly calculate non-zero elements without building a full matrix as shown in the following subsections.

3.2. Special Formulation for Chain Structure

For a chain structure such as hair, we can take advantage of its linear topology and serial indexing to create a symmetric tridiagonal matrix formulation. In Fig. 1, vertices and constraints are indexed in one direction in the serial manner. Also in case of hair or fur, we choose the first vertex as an

attachment point and assign infinite mass.

$$-\frac{1}{m_i}\frac{\partial C_i}{\partial \mathbf{p}_i}\frac{\partial C_{i-1}}{\partial \mathbf{p}_{i-1}}\Delta\lambda'_{i-1} + \left[\frac{1}{m_i}\left(\frac{\partial C_i}{\partial \mathbf{p}_i}\right)^2 + \frac{1}{m_{i+1}}\left(-\frac{\partial C_i}{\partial \mathbf{p}_i}\right)^2\right]\Delta\lambda'_i - \frac{1}{m_{i+1}}\frac{\partial C_i}{\partial \mathbf{p}_i}\frac{\partial C_{i+1}}{\partial \mathbf{p}_{i+1}}\Delta\lambda'_{i+1} = C_i \quad (5)$$

Eqn. 5 shows one equation from Eqn. 3 for a linear structure. In case C_0 or C_{n-2}, we can simply drop the term for $\Delta\lambda_{-1}$ or $\Delta\lambda_{n-1}$. From Eqn. 5, it is clear that the linear equation forms a symmetric and tridiagonal matrix. The first term formulates subdiagonal, the middle one does diagonal and the third one does superdiagonal.

$$\Delta\mathbf{p}_i = -\frac{1}{m_i}\left(-\frac{\partial C_{i-1}}{\partial \mathbf{p}_{i-1}}\Delta\lambda'_{i-1} + \frac{\partial C_i}{\partial \mathbf{p}_i}\Delta\lambda'_i\right) \quad (6)$$

Eqn. 6 shows one equation from Eqn. 4.

3.3. Matrix-free Formulation

We define the constraint as $C_i = \|\mathbf{p}_i - \mathbf{p}_{i+1}\| - r_i$, r_i is a rest length, and the constraint gradient can be defined as $\frac{\partial C_i}{\partial \mathbf{p}_i} = -\frac{\partial C_i}{\partial \mathbf{p}_{i+1}} = \frac{\mathbf{p}_i - \mathbf{p}_{i+1}}{\|\mathbf{p}_i - \mathbf{p}_{i+1}\|} = \mathbf{n}_i$. To simplify the formulation, we set zero inverse mass to the fixed vertices and one to the free ones. It is actually not a matter to assign any mass value as long as the all free vertices have the equal mass. We can use other constraint definitions such as quadratic length but we can get a benefit from our formulation because the constraint gradient becomes a normalized edge vector and the coefficients of $\Delta\lambda'_i$ become constant. Below is a simplified equation from Eqn. 5 by using our constraint and mass definitions.

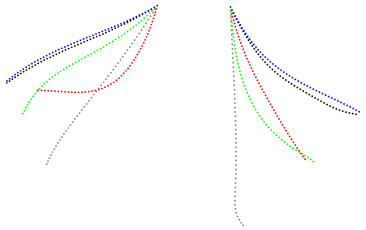
$$-\mathbf{n}_{i-1}\mathbf{n}_i\Delta\lambda'_{i-1} + 2\Delta\lambda'_i - \mathbf{n}_i\mathbf{n}_{i+1}\Delta\lambda'_{i+1} = (\|\mathbf{p}_i - \mathbf{p}_{i+1}\| - r_i) \quad (7)$$

In hair or fur simulation, we define the first vertex is fixed and other vertices are free. The first constraint equation can be defined as $\Delta\lambda'_0 - \mathbf{n}_0\mathbf{n}_1\Delta\lambda'_1 = (\|\mathbf{p}_0 - \mathbf{p}_1\| - r_0)$ by setting $\frac{1}{m_0} = 0$. The last constraint equation is $-\mathbf{n}_{n-3}\mathbf{n}_{n-2}\Delta\lambda'_{n-3} + 2\Delta\lambda'_{n-2} = (\|\mathbf{p}_{n-2} - \mathbf{p}_{n-1}\| - r_{n-2})$.

The matrix has a constant diagonal and is symmetric. Therefore, the superdiagonal and subdiagonal are the same and their element is basically the dot product of two adjacent normalized edge vectors. Therefore, we do not need to construct a matrix explicitly and it is easy to calculate the non-zero elements by simple dot products.

3.4. Solving Linear System

In Fig. 1, there are $n-1$ equations and $n-1$ unknowns. This linear equations can be solved by any solver. For a cloth simulation, we could use such as a general direct solver but the computational overhead is high compared to position-based solver. This is one of the reasons why the position-based solver is widely used in real-time applications.



(a) (b)

Figure 2: Screenshots from hair strand simulations. Blue, red, gray, green, and black strands are simulated using TMF, DFTL and PBD with 2, 5, 40 iterations respectively.

However, the coefficient matrix for our problem is sparse and only contains diagonal, subdiagonal and superdiagonal elements as we have shown above. Furthermore, the matrix is diagonally dominant because subdiagonal and superdiagonal elements are dot products of two unit vectors. For this matrix, we can use tridiagonal matrix algorithm (Thomas algorithm) which is a direct solver consists of a forward sweep

$$c'_i = \frac{c_i}{b_i - c'_{i-1}a_i} \quad (8)$$

$$d'_i = \frac{d_i - d'_{i-1}a_i}{b_i - c'_{i-1}a_i} \quad (9)$$

and backward sweep

$$x_i = d'_i - c'_i x_{i+1} \quad (10)$$

where a_i, b_i, c_i are subdiagonal, diagonal, superdiagonal elements [PTVF07].

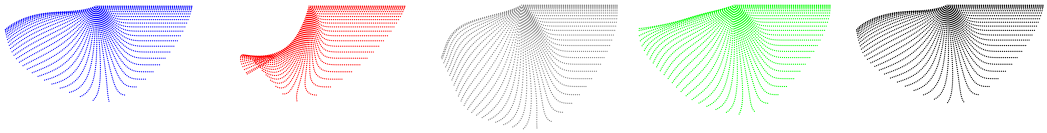
This algorithm is suited for real-time purpose because it is simple to implement, and it does not require any additional memory storage. Therefore, it is even possible to implement on the GPU as we are going to show later.

3.5. Update Positions

After solving the linear equations, we can get position corrections from Eqn. 6. Here, we can also take advantage of the simplicity of linear topology and our constraint definition as below.

$$\Delta\mathbf{p}_i = \mathbf{n}_{i-1}\Delta\lambda'_{i-1} - \mathbf{n}_i\Delta\lambda'_i \quad (11)$$

By this simple position update formulation, we can avoid constructing a gradient matrix $\nabla\mathbf{C}(\mathbf{P})$ explicitly and solving Eqn. 4.



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

Figure 3: Trajectories of the hair strand simulations using (a) TMF, (b)DFTL and (b), (c), (d) PBD with 2, 5, 40 iterations, respectively.

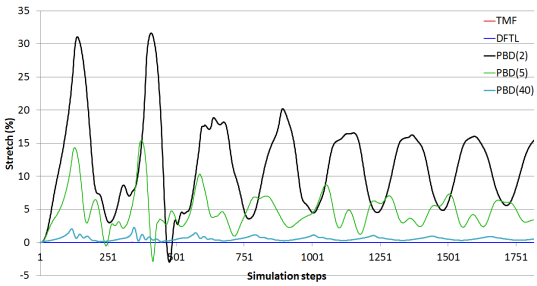


Figure 4: Comparison of stretch of the hair simulations.

TMF	DFTL	PBD(2)	PBD(5)	PBD(40)
0.067	0.000	31.62	15.38	2.26

Table 1: Maximum stretch (%) of the hair simulations.

3.6. Implementation

The proposed algorithm is implemented on the CPU and the GPU. For the GPU implementation, OpenCL is used and each work item calculates a hair strand, i.e., it builds the coefficients and updates all the vertices in the strand. Although the computation of a single strand is serialized, we are still able to exploit the entire GPU for a simulation where there are a lot of strands as examples shown in Fig. 5 and 6.

4. Results

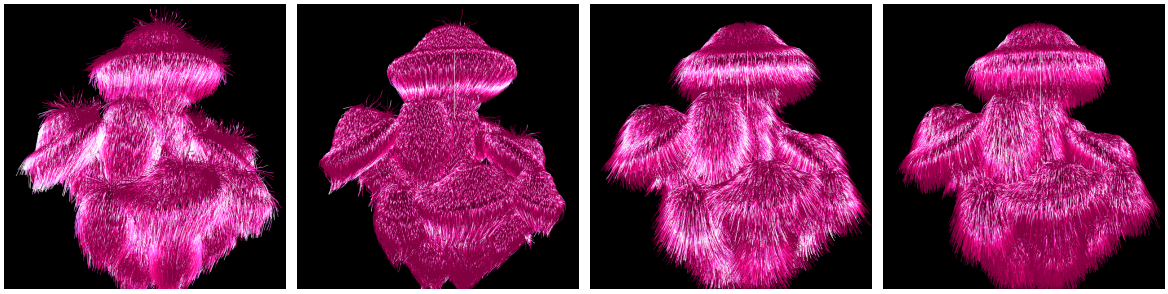
To evaluate the inextensibility of the method, a hair strand with 51 vertices is simulated by the proposed method (TMF), Dynamic Follow-The-Leader (DFTL) and position-based solver with 2, 5 and 40 iterations for distance constraints. The left most vertex at the beginning of the simulation is set infinite mass to fix it. Simulation results are shown in Fig. 2 and trajectories for vertices in those simulations are shown in Fig. 3 in which stretch of position based solver is apparent. We measured the total length of the strand for the simulation and compared the ratio of stretch to the original strand length (Fig. 4). Even the solution of position-based solver with 40 iterations are stretching more than 2% at the first swing. However, we can see from the graph that the stretch of the proposed method is almost visually unnoticeable as DFTL. Actually, the maximum stretch of the proposed method is less than 0.1% (Table 1). DFTL shows the best inextensibility but the difference between TMF and DFTL is almost negligible.

We can also see from Fig. 3 that the hair strand simulated with TMF reaches as high as PBD with 40 iterations after a swing. This is because our method does not generate any artificial damping and enforces the constraints tightly. This is the biggest difference from the method by DFTL.

As for the computational cost, we cannot directly compare the proposed method to position-based dynamics because the proposed method does not solve constraints in the same way to position-based dynamics. However, our method only iterates the constraints twice in the solve of linear equations. Iterating over constraints twice is almost equivalent to using two iterations for position-based solver. Therefore, we could say that the computational cost or complexity of the proposed method is equivalent to position-based solver with two iterations whose convergence is very poor as we show in Fig. 4.

Regarding stability of TMF compared to DFTL or PBD, it is less stable and dependent to relatively small timestep due to the linearization of constraint system. It may be possible to formulate the ill matrix if the two adjacent edges are perpendicular. However it is easy to find those cases and in practical case, it does not happen much due to the bending constraints.

Fig. 5 shows screenshots from a simulation using our GPU implementation running on a Radeon HD 7970 GPU. There are 100K hair strands and 600K simulated vertices which are simulated and rendered at 60fps. There are distance constraints connecting adjacent two vertices which are solved by the proposed method, and bending constraints solved by position-based solver. In a step of the simulation, external force is applied and positions are integrated to calculate non-constrainted positions. Then it applies bending constraints 8 times and distance constraints are solved once by the tridiagonal matrix solver. Those are iterated twice in a step. A solve of the tridiagonal matrix solver took 1.5ms which is about 16% of the entire simulation time. As we iterate constraint solvers two times in a simulation step, 3ms is spent to enforce distance constraints. As the computation of



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

Figure 5: Mushrooms with 100K hair strands simulated on the GPU.

the proposed method was fast enough, we did not optimize the kernel further. However, we found out that the current implementation uses a lot of registers which lowers the occupancy of the GPU. Offloading the register pressure using other memories such as local data store is a future work. The bottleneck of the simulation is solving bending constraints using position-based solver which requires many iterations to enforce a high bending resistance.

Computation time is linear to the number of constraints in the scene. A scene with 200K hair strands is simulated as shown in Fig. 6 which runs at 30fps.

5. Conclusion

This paper proposes a method to simulate inextensible hair and fur using tridiagonal matrix formulation in which distance constraints are formulated as a linear system. Although the proposed method constructs and solves linear equations, it does not require to assemble a full matrix which is beneficial for low memory usage and fast computation. We propose a matrix-free formulation in which a few nonzero elements are calculated directly from the topology of constraints. Then we show that a computationally inexpensive direct solver can be used to solve the linear equation. Comparing stretch of the proposed constraint solver to the position-based solvers with different number of iterations, we show that error from the proposed method is much smaller than those from the position-based solvers and very close to one from the Dynamic Follow-The-Leader method.

For future work, we are planning to extend this formulation to higher dimensional manifolds such as cloth or soft volume mesh by decomposing them into linear chain structures.

References

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 594–603. 2

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. 1

[CCK05] CHOE B., CHOI M. G., KO H.-S.: Simulating complex hair with robust collision handling. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 153–160. 1, 2

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), to appear. 2

[HDB96] HOUSE D., DEVAUL R. W., BREEN D. E.: Towards simulating cloth dynamics using interacting particles. *International Journal of Clothing Science and Technology 8* (1996), 75–94. 2

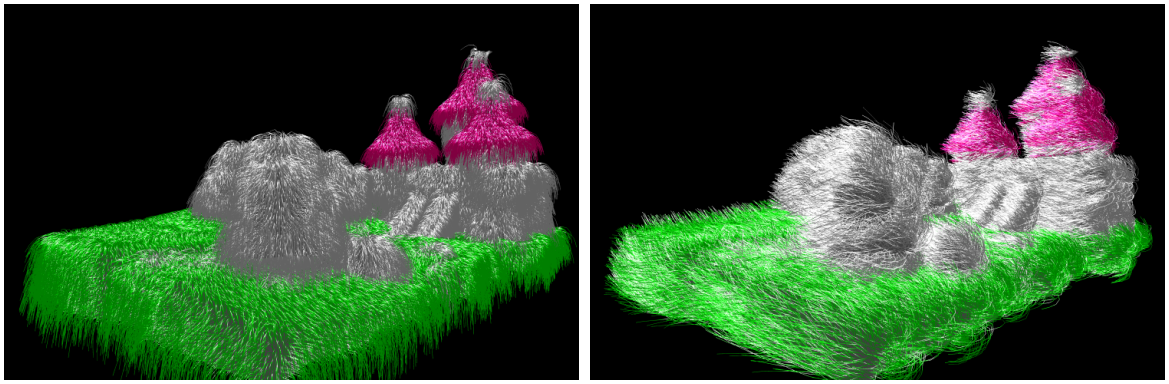
[HH12] HAN D., HARADA T.: Real-time Hair Simulation with Efficient Hair Style Preservation. In *VRIPHYS 12: 9th Workshop on Virtual Reality Interactions and Physical Simulations* (2012), pp. 45–51. 2

[HMT01] HADAP S., MAGNENAT-THALMANN N.: Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3 (2001), 329–338. 2

[IMP*13] IBEN H., MEYER M., PETROVIC L., SOARES O., ANDERSON J., WITKIN A.: Artistic simulation of curly hair. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, ACM, pp. 63–71. 2

[KCMF12] KIM T.-Y., CHENTANEZ N., MÜLLER-FISCHER M.: Long range attachments - a method to simulate inextensible clothing in computer games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2012), SCA '12, Eurographics Association, pp. 305–310. 2

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. 1, 2



(a) (b)

Figure 6: Furry castle with 200K hair strands simulated on the GPU running at 30fps.

[MKC12] MÜLLER M., KIM T.-Y., CHENTANEZ N.: Fast simulation of inextensible hair and fur. In *VRIPHYS* (2012), Bender J., Kuijper A., Fellner D. W., Guérin E., (Eds.), Eurographics Association, pp. 39–44. 1, 2

[MSW*09] MCADAMS A., SELLE A., WARD K., SIFAKIS E., TERAN J.: Detail preserving continuum simulation of straight hair. *ACM Trans. Graph. 28*, 3 (2009). 2

[Mül08] MÜLLER M.: Hierarchical position based dynamics. In *VRIPHYS* (2008), pp. 1–10. 2

[Pro96] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *In Graphics Interface* (1996), pp. 147–154. 2

[PTVF07] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes 3rd Edition: The Art of Scientific Computing. Section 2.4.* Cambridge University Press, New York, NY, USA, 2007. 3

[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. 1

[RKN10] RUNGJIRATANANON W., KANAMORI Y., NISHITA T.: Chain shape matching for simulating complex hairstyles. *Comput. Graph. Forum* (2010), 2438–2446. 2

[Sch06] SCHENK OLAF G. K.: On fast factorization pivoting methods for sparse symmetric indefinite systems. *ETNA. Electronic Transactions on Numerical Analysis [electronic only] 23* (2006), 158–179. 2

[SH10] SPILLMANN J., HARDERS M.: Inextensible elastic rods with torsional friction based on lagrange multipliers. *Journal of Visualization and Computer Animation 21*, 6 (2010), 561–572. 2

[SLF08] SELLE A., LENTINE M., FEDKIW R.: A mass spring model for hair simulation. In *ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 64:1–64:11. 1, 2

[WBK*07] 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 13*, 2 (2007), 213–234. 2

[WL03] WARD K., LIN M.: Adaptive grouping and subdivision for simulating hair dynamics. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on* (oct. 2003), pp. 234 – 243. 2