Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

Fast Simulation of Inextensible Hair and Fur

M. Müller T.Y. Kim N. Chentanez

Nvidia PhysX Research

Abstract

In this short paper we focus on the fast simulation of hair and fur on animated characters. While it is common in films to simulate single hair strands on virtual humans and on furry animals, those features are either not present on characters in computer games or modeled with simplified textured meshes. The main difficulty of simulating hair in real time applications is the sheer number of hair strands and the fact that each hair is inextensible. Keeping thousands of deformable objects from being stretched is computationally expensive. In this paper, we present a robust method for simulating hair and fur that guarantees inextensiblity with a single iteration per frame. For an iteration count this low, existing methods either become unstable or introduce a substantial amount of stretching. Our method is geometric in nature and able to simulate thousands of inextensible hair strands in real time.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically Based Modeling; Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation and Virtual Reality

1. Introduction

While simulating hair and fur is common in the movie industry, those features are rarely seen in today's computer games. Simplified approaches represent human hair as a relatively small set of mesh stripes with alpha textures. While this approach yields plausible results for human hair to a certain extent, it is not well suited for simulating characters that are covered with fur.

One of the reasons for using this simple approach is that there is a lack of simulation methods that are fast and robust enough to simulate thousands of hair strands in real time. Besides the large number of hair strands, one of the main challenges is that each hair is perceived as inextensible. Guaranteeing zero stretch in simulations of deformable objects is a non-trivial problem.

One method to simulate inextensible ropes is to use the angles between segments as the degrees of freedom instead of the positions of mass points. In this subspace of spatial configurations, a rope always keeps its rest length. However, using generalized coordinates in the presence of colliding objects and in over-constrained situations becomes a difficult problem and its solution expensive.

With the Euclidean coordinates of the mass points or finite elements as the degrees of freedom, inextensibility can be achieved in the limit by using internal coupling with infinite stiffness. However, simply increasing the stiffness constants introduces stability and damping issues.

Instead of using spring-like forces, one can solve for constraint forces that yield velocities which keep the mass points on trajectories that do not change the constraint functions. One problem with this approach is that the mass points can drift away from the conserving paths, especially when large time steps are used.

Therefore, inextensibility is typically achieved by limiting strain geometrically after the dynamics solver has updated the positions of the mass points at each time step. In the case of a mass spring system, the strain limiting step moves the mass points such that all the springs are not extended by more than a given factor. Finding such positions is a global, non-linear problem which is typically solved by either using multiple global Newton-Raphson solves [GHF*07] or multiple passes through all the constraints [MHR06].

In this paper we present a simulation method that takes only one iteration per visible frame while guaranteeing zero-stretch. Our method extends a technique called *Follow The Leader* which as previously been used for quasy-static simulations only [BLM04].

Being geometric, our method is not as accurate as physi-

DOI: 10.2312/PE/vriphys/vriphys12/039-044

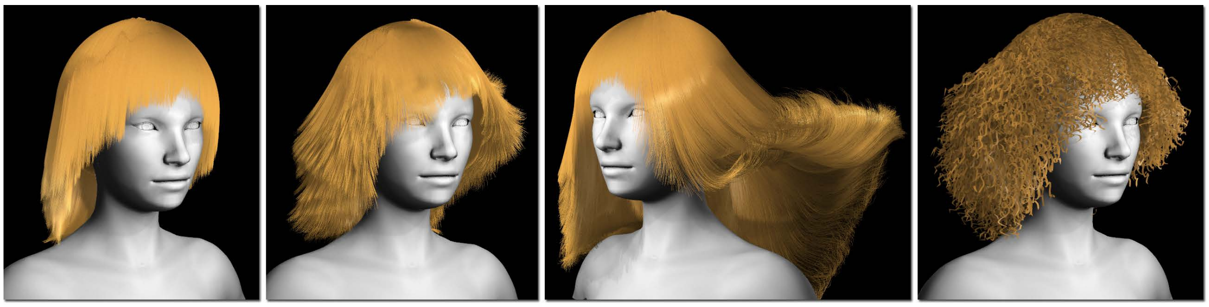

Figure 1: *Our method allows the simulation of every hair strand in real time. From left to right: 47k hairs simulated at 25 fps including rendering and hair-hair repulsion. Long hair composed of 1.9m particles at 8 fps. Curly hair using visualization post-processing.*

cally based techniques in that it introduces a certain amount of artificial damping. In most situations we encountered, this is not problematic because objects in the real world are quite heavily damped in general. Our target scenario is adding hair and fur to animated characters. Those actors constantly add energy to the system when they move.

2. Related Work

The simulation of one dimensional elastic solids such as hair and rods has been studied extensively in computer graphics. For an excellent survey of the field we recommend [WBK*07] and [HCB*07]. More recent developments include the study of discrete elastic rods [BWR*08] and viscous threads [BAV*10] which treat the centerline as dynamic and the material frame as static. Spillman and Teschner [ST07,ST08] represent material frames with quaternions and couple them to the centerline with penalty forces. Kubiak et al. [KPGF07] use PBD to simulate threads in knot tying applications. A more analytical approach was taken by Theetten et al. [TGAB08] who use dynamic splines for the simulation. Selle et al. [SLF08] propose to use altitude springs to handle torsion in hair simulation and the semi-implicit discretization of springs to make the simulation unconditionally stable. Bertails [Ber09] devised a linear time algorithm to simulate piecewise helical rods. Sueda et al. [SJLP11] propose the use of reduced degree of freedom Eulerian nodes to handle the simulation of ropes and cables in highly constrained scenarios.

Since the explicit handling of hair-hair interaction is expensive, [PHA05] proposed to use a regular background grid to compute a hair density field which is used to approximate mutual hair repulsion and friction. Our method for hair-hair interaction is based on this idea. A similar approach was used by McAdams et al. [MSW*09]. They handle the bulk interaction and volume preservation of hairs using an Eulerian grid.

Often it is not necessary to simulate every individual hair. Chang et al. [CJY02] proposed to only simulate a subset of hairs called *key hairs* and interpolated all other hair shapes from this subset. This same approach was used by [Tar08] in a real-time hair rendering framework. One of the drawbacks of the method is that interpolated hairs do not properly collide against the character. Also, smooth hair tends to look clumpy depending on the number of key hairs used. In contrast, our method allows the simulation of every individual hair in real time which brings a new level of realism to interactive applications such as computer games.

3. Simulation Method

We model a single hair by a chain of particles that is attached at one end (see Figure 2). Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be the positions of the particles with particle 1 being attached and let us assume that the rest distances between adjacent particles are all l_0. Starting with positions that violate the distance constraints we want to move the particles such that all the constraints are satisfied but with the restriction that we are only allowed to iterate through all the particles once.

3.1. Static Follow-The-Leader (FTL)

One way to do this is to process the particles in the order from 2 to n. Particle 2 has to be on a sphere with radius l_0 around particle 1. A natural choice for its position is to choose the point on the sphere that is closest to the original position $\mathbf{x}_2$, i.e. to move it in the direction of particle 1. In case of a collision, the two remaining degrees of freedom of particle 2 within the sphere can be used to resolve the collision simultaneously. In this case, one chooses the position closest to $\mathbf{x}_2$ that resolves the collision. Once the new position of particle 2 is determined, the algorithm continues with particle 3 preforming the same steps as before with particle 2 taking the role of particle 1. Figure 2(a) shows the algorithm for a chain of four particles.

This algorithm is called *Follow The Leader* (FTL). In the field of physically based simulation it was used by [BLM04]

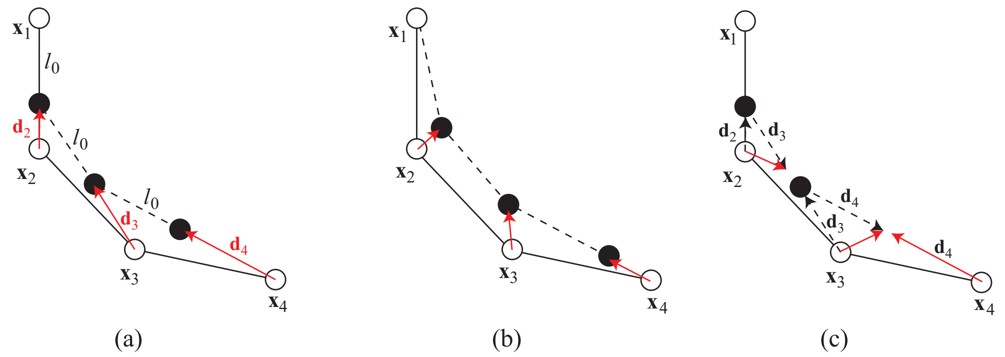

Figure 2: *(a) FTL projection: the white points show the positions of the particles before projection with the topmost fixed. Each particle is moved towards its predecessor to enforce their mutual distance to be l_0. The red arrow $\mathbf{d}_i$ is the resulting velocity corrections of particle i. (b) Projection using multiple iterations of PBD and the resulting velocity corrections in red. The velocity corrections are tilted to the right due to the pull of the subsequent chain. (c) DFTL projection: the velocity correction of particle i is the difference $\mathbf{d}_i$ - $\mathbf{d}_{i+1}$. This vector is tilted to the right as well but more than in the case of PBD which introduces damping.*

for the quasy static simulation of knot tying. One of the main reasons why it has not become popular in the field is that making it work in a dynamic simulation is challenging. It is this problem that we will investigate next.

3.2. Dynamic Follow-The-Leader (DFTL)

For a dynamic simulation, the particles need to store and update velocities along with the positions. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the velocities of the particles. Since we manipulate positions to drive the simulation we need a way to derive the velocity updates from the position updates. Position Based Dynamics (PBD) [MHR06] provides such a way. The algorithm to perform one time step of PBD has the following simple structure

$$\mathbf{p} \leftarrow \mathbf{x} + \Delta t\, \mathbf{v} + \Delta t^2 \mathbf{f} \tag{1}$$
$$\mathbf{p} \leftarrow \text{SolveConstraints}(\mathbf{p}) \tag{2}$$
$$\mathbf{v} \leftarrow \frac{\mathbf{p} - \mathbf{x}}{\Delta t} \tag{3}$$
$$\mathbf{x} \leftarrow \mathbf{p}, \tag{4}$$

where Δt is the time step size and $\mathbf{f}$ external forces. To make FTL dynamic we could simply perform static FTL inside the SolverConstraints() method. This way, we would still only need one iteration per time step because the four steps can be performed together in a single pass.

However, this straight forward method yields the strange behavior shown in Figure 3 and in the accompanying video for a horizontal chain attached on the left and falling under gravity. To understand why this simple approach does not work we have to have a closer look at the physical system we are simulating. In traditional PBD to solve a distance constraint between two particles, they are moved towards or away from each other by distances proportional to their inverse masses. Therefore, moving just the second particle corresponds to the situation in which the first particle has infinite mass. In general, by using FTL as the projection step in a dynamic simulation, a system with masses $m, sm, s^2m, \ldots s^{n-1}m$ with $s \rightarrow 0$ is simulated – independent of the integration method.

We have not found a discussion of this interesting, physically impossible system in the literature but came to the following intuition about its behavior. At a given point in time as shown in any image of Figure 3, a series of particles starting from the attachment downwards has come to rest due to damping. One particle below them contains all the current kinetic energy of the system and swings around all the particles below it effortlessly because the sum of all the masses of the tail is infinitely smaller than its own mass. Moreover, none of its motion is propagated to the parents because they, have infinitely more mass.

3.3. Velocity Correction

Deriving the dynamic behavior of a chain of particles with equal masses after an FTL projection step is surprisingly difficult because the extremely uneven mass distribution is deeply inherent to the FTL projection. After investigating various methods that all failed, we found a very simple solution. It completely hides the uneven mass distribution at the price of introducing some numerical damping.

Let $\mathbf{d}_i$ be the correction vector computed by FTL for particle i. When considering constraint $(i-1, i)$, FTL updates the positions of particles $i-1$ and i as

$$\mathbf{p}_{i-1} \leftarrow \mathbf{p}_{i-1} \text{ (no update)} \tag{5}$$
$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \mathbf{d}_i. \tag{6}$$

To generate the behavior of two particles with the same mass

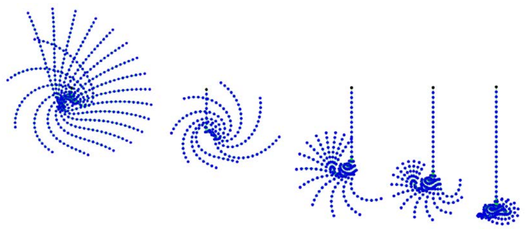

Figure 3: *Follow The Leader projection yields the dynamic behavior of a chain of particles each of which has infinitely more mass then its successor. The figure shows a simulation of such a chain starting as a horizontal line attached at the left end and falling under gravity. Each image corresponds to a different point in time and shows a set of overlayed successive frames.*

we would have to move particle $i-1$ as well by the same amount as

$$\mathbf{p}_{i-1} \leftarrow \mathbf{p}_{i-1} - \mathbf{d}_i \quad (7)$$

$$\mathbf{p}_i \leftarrow \mathbf{p}_i \quad + \mathbf{d}_i \quad (8)$$

in the update step of particle i. However, this would violate the constraint between particle $i-2$ and particle $i-1$. The idea to circumvent this problem is to use this symmetric position of particle $i-1$ only to compute the new velocity in Eq. (3) at the end of the time step, not for the position of the particle. We, thus, modify Eq. (3) as follows:

$$\mathbf{v}_i \leftarrow \frac{\mathbf{p}_i - \mathbf{x}_i}{\Delta t} + s_{\text{damping}} \frac{-\mathbf{d}_{i+1}}{\Delta t}. \quad (9)$$

The velocities derived from FTL without corrections are shown in 2(a) in red. 2(b) shows the velocities resulting from multiple iterations of PBD - the physically correct velocities. These are tilted more to the right due to the pull of the subsequent chain. The corrected FTL velocities shown in 2(c) are tilted to the right as well but more against the pulling direction which introduces damping.

Since Eq. (9) is our main contribution, we would like to emphasize that this contribution is non-trivial. As mentioned above, we have experimented with various ways to derive velocities that hide the uneven mass distribution of FTL. The solution presented here is the only one we found. For instance, at first glance, it seems more natural to introduce a factor of $\frac{1}{2}$ in Eqs. (7) and (8). However, not compensating for the entire displacement of FTL still results in the behavior shown in Fig. 3. We also experimented with deriving the velocities in a completely different way, decoupled from the FTL projections of the positions and did not find a solution yielding natural behavior.

In Eq. (9) we introduced the scalar parameter $s_{\text{damping}} \in [0,1]$. For $s_{\text{damping}} = 1$ the uneven masses are completely compensated for but with the introduction of numerical damping (see Figure 4 left). For s_{damping} smaller but close to 1, damping is reduced while the artifact of the uneven masses is still hardly noticeable (see Figure 4 right and the accompanying video). Reducing s_{damping} is a simple and computationally cheap way to reduce numerical damping.

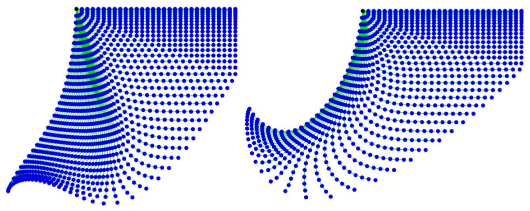

Figure 4: *Rope simulation with velocity correction. Left: Correction scale $s_{damping} = 1$. Right: Correction scale $s_{damping} = 0.9$.*

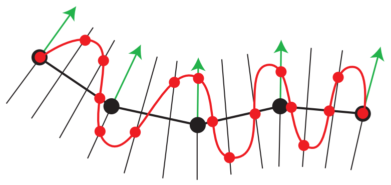

Figure 5: *To simulate curly hair, we generate separate vertices for rendering (red) by subdividing segments and offset the interpolation particle positions along the particle normals (green).*

3.4. Curly Hair

To simulate curly hair we do not modify the simulation model but create a separate chain of vertices for rendering only (see Figure 5). To this end, we transport a normal down the hair in FTL style. The first normal is defined by the attachment. To compute the normal of particle i we make the normal of particle $i-1$ perpendicular to each adjacent segment of particle i by removing the parallel components. The normal is then the normalized average of those two normals. The bi-normals can be computed as the cross product with the hair direction. For subdivision points on the segments we simply linearly interpolate and normalize the frames. The visual vertices are then computed within the plane spanned by normals and bi-normals. We use a spiral with decreasing frequency along the hair. To get realistic behavior we set the hair thickness for character collision to the radius of the spiral and increase the hair-hair repulsion and friction forces as described in the next Section. (see Figure 1 right).

3.5. Hair-Hair Interaction

With an increasing number of hairs, handling hair-hair interactions and collisions with the character's skin become the bottleneck. To handle hair-hair friction and repulsion, we use the method proposed by Petrovic et al [PHA05]. At each time step we first compute a particle density field on a regular background grid. Each particle adds its tri-linear interpolation weights to the density values of the 8 nodes of the surrounding cell. It also adds its velocity multiplied by the tri-linear interpolation weights to the velocities of the 8 nodes. Dividing the velocities by the densities stored at the grid nodes yields an averaged velocity field that can be used to imitate friction.

For each particle, we interpolate this velocity field at the particle's position. We then replace the particle's velocity with a weighted average of its own velocity and the velocity interpolated from the grid at each time step.

$$\mathbf{v} \leftarrow (1 - s_{\text{friction}})\mathbf{v} + s_{\text{friction}}\mathbf{v}_{\text{grid}} \tag{10}$$

By tuning the weight s_{friction} we can control the amount of friction.

Hair-hair repulsion is needed to get volumetric hairstyles and to get the impression of finite hair thickness. To this end, we compute the normalized gradient $\mathbf{g} = \nabla\rho/|\nabla\rho|$ of the density field at the particle's location as [PHA05] but use a different way to update the velocity:

$$\mathbf{v} \leftarrow \mathbf{v} + s_{\text{repulsion}}\mathbf{g}/\Delta t. \tag{11}$$

Both velocity correction steps are executed after PBD integration.

3.6. Collision with the Character

When the hair is not too long as is the case for furry characters, one can ignore character collision without introducing disturbing visual artifacts when hair-hair repulsion is turned on as Figure 7 shows. When the color of the fur and of the underlying skin are similar, it is even possible to get away with neither hair-hair nor hair-body interaction as the first image shows.

In the case of longer human hair, collision with the character is essential. For the simulations shown in Figure 1 we used a simplified collision volume composed of 8 ellipsoids.

4. Results

Figure 6 shows a comparison of dynamic FTL (blue), PBD (green) and symplectic Euler (red). A rope composed of 30 particles is dragged around by the user at the top. The time step size, particle masses, segment lengths and gravitational acceleration are $0.01s$, $0.01kg$, $0.02m$ and $-10m/s^2$, respectively. In a first experiment (left) we let all three methods spend about same amount of time, i.e. 2 iterations for PBD and 2 sub-steps for symplectic Euler. The maximum stiffness allowed to keep the latter stable was $k = 100N/m$. Both PBD and Euler show a substantial amount of stretching. In a second experiment we adjusted the parameters such that all three ropes showed similar behavior. To achieve this 25

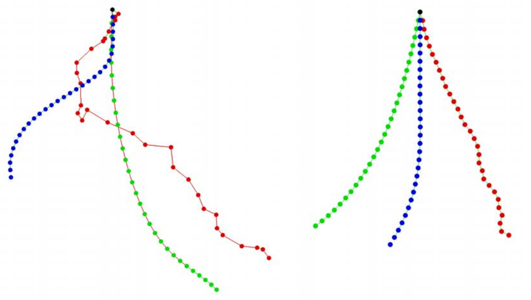

Figure 6: *Screen shots of an animation in which the top particle of a rope is dragged around using three simulation methods: dynamic FTL (blue), PBD (green) and symplectic Euler (red). Left: all three methods spend the same amount of time per frame. Right: Time spend is adjusted to yield similar results.*

PBD iterations where needed. For Euler integration we had to increase the stiffness to $k = 3000N/m$ which could only be simulated stably with 20 substeps.

To demonstrate the performance and visual plausibility of our method in realistic scenarios, we used it to simulate both fur and human hair. Both scenes shown in Figures 1 and 7 run at interactive rates with the parallelized CUDA version of our algorithm on an NVIDIA GeForce GTX 480 GPU.

The short hair shown in the first two images of Figures 1 is composed of 47k hairs and 776k simulated particles. Turning hair-hair repulsion on and off yields different hair styles. The simulation including rendering and collision against 8 ellipsoids runs at 25 fps. The long hair shown in the third image of Figure 1 also has 47k hair strands but 1.9m particles. The scene still runs at interactive 8 fps. The fourth image shows the result of applying our simple method to create a curly hair style. Since we only use 2k and 38k particles, the simulation is much faster and runs at 80fps.

Figure 7 shows a furry monster. The fur is modeled with 100k hair strands and 684k particles. As mentioned previously, collision with the character is ignored because hair-hair repulsion makes the hairs move away from the skin. The third image shows the interaction with a hair blower on the head. This scenario would be difficult to handle with the key hair approach because the hairs get separated to point into different directions at arbitrary locations.

5. Conclusion

We have presented a method to simulate the dynamic behavior of inextensible hair and fur which guarantees zero-stretch in a single iteration per visual frame. Being geometric and approximative, our method introduces a certain amount of numerical damping. However, we did not find this to be a significant drawback because characters constantly add en-

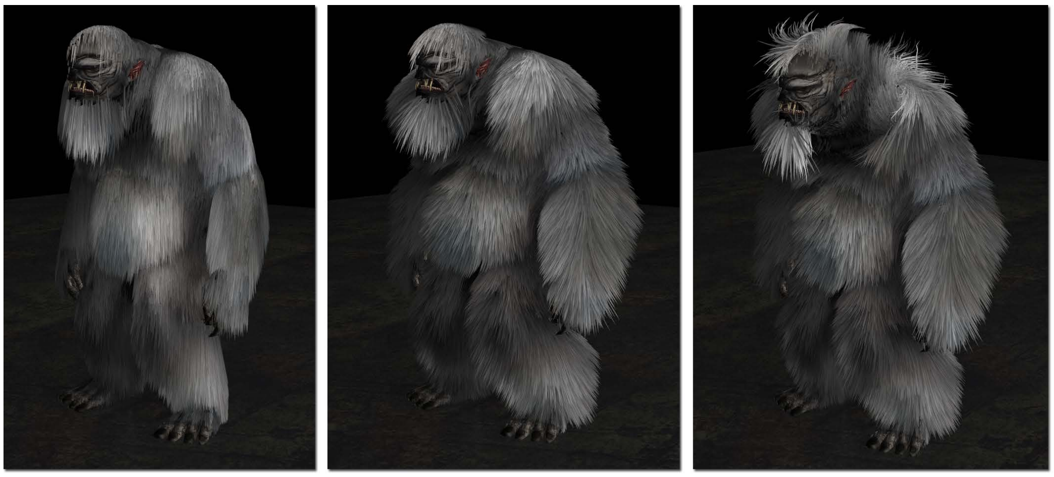

Figure 7: *In the case of fur, hair-hair repulsion allows to ignore hair-character collision without the introduction of disturbing artifacts. The third image shows the interaction with a hair blower on the head.*

ergy when they are in motion and objects in the real world are significantly damped. We also proposed a simple method to reduce artificial damping. As the results show, our method makes possible the simulation of tens of thousands of hairs in real-time including character collision handling and hair-hair interaction.

References

[BAV*10] Bergou M., Audoly B., Vouga E., Wardetzky M., Grinspun E.: Discrete viscous threads. *ACM Trans. Graph. 29* (July 2010), 116:1–116:10. 2

[Ber09] Bertails F.: Linear time super-helices. *Comput. Graph. Forum 28*, 2 (Apriö 2009), 417–426. 2

[BLM04] Brown J., Latombe J.-C., Montgomery K.: Real-time knot-tying simulation. *Vis. Comput. 20* (May 2004), 165–179. 1, 2

[BWR*08] Bergou M., Wardetzky M., Robinson S., Audoly B., Grinspun E.: Discrete elastic rods. *ACM Trans. Graph. 27* (August 2008), 63:1–63:12. 2

[CJY02] Chang J. T., Jin J., Yu Y.: A practical model for hair mutual interactions. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 73–80. 2

[GHF*07] Goldenthal R., Harmon D., Fattal R., Bercovier M., Grinspun E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26* (July 2007). 1

[HCB*07] Hadap S., Cani M.-P., Bertails F., Lin M. C., Ward K., Marschner S. R., Kim T.-Y., Kacic-Alesic Z.: *Strands and Hair: Modeling, Animation, and Rendering*, vol. 33. ACM SIGGRAPH Course Notes, Mai 2007. Award: Best Course Notes for a New Course. 153 pages. 2

[KPGF07] Kubiak B., Pietroni N., Ganovelli F., Fratarcangeli M.: A robust method for real-time thread simulation. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2007), VRST '07, ACM, pp. 85–88. 2

[MHR06] Müller M., Hennix B. H. M., Ratcliff J.: Position based dynamics. *Proceedings of Virtual Reality Interactions and Physical Simulations* (2006), 71–80. 1, 3

[MSW*09] McAdams A., Selle A., Ward K., Sifakis E., Teran J.: Detail preserving continuum simulation of straight hair. *ACM Trans. Graph. 28* (July 2009), 62:1–62:6. 2

[PHA05] Petrovic L., Henne M., Anderson J.: Volumetric methods for simulation and rendering of hair. In *Tech. rep., Pixar Animation Studios* (2005). 2, 5

[SJLP11] Sueda S., Jones G. L., Levin D. I. W., Pai D. K.: Large-scale dynamic simulation of highly constrained strands. *ACM Trans. Graph. 30* (Aug. 2011), 39:1–39:10. 2

[SLF08] Selle A., Lentine M., Fedkiw R.: A mass spring model for hair simulation. *ACM Trans. Graph. 27* (August 2008), 64:1–64:11. 2

[ST07] Spillmann J., Teschner M.: M.: Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *In Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 63–72. 2

[ST08] Spillmann J., Teschner M.: An adaptive contact model for the robust simulation of knots. *Comput. Graph. Forum 27*, 2 (2008), 497–506. 2

[Tar08] Tariq S.: Real-time hair rendering on the gpu. In *ACM SIGGRAPH 2008 talks* (2008). 2

[TGAB08] Theetten A., Grisoni L., Andriot C., Barsky B.: Geometrically exact dynamic splines. *Comput. Aided Des. 40* (January 2008), 35–48. 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* (March 2007), 213–234. 2