Position-based Shape Constraint for Real-time Hair Simulation

Matheus de Moraes Cavazotti
Universidade Federal do Paraná
Curitiba, Brazil
Email: mcavazotti@gmail.com

André Luiz Pires Guedes
Universidade Federal do Paraná
Curitiba, Brazil
Email: andre@inf.ufpr.br

***Abstract*—In this work we present a method to simulate curly hair with its spring-like characteristics in real-time, while maintaining the strand inextensibility. By using the position-based simulation framework and expanding on previous works on the same topic, a new constraint is devised to keep the curls shape. We share our results and stablish parallels with other related works.**

I. Introduction

This work is fruit of the curiosity to understand more deeply how real-time hair simulation works. What began as a personal project eventually evolved into a Masters dissertation, and as we studied the topic, the relevance of our research became even more apparent.

Whenever there is a human character or some other fur-covered creature portrayed in a computer generated image, there is the need to represent visually what hair looks like. That's not a trivial task, since there are more than 100.000 strands of hair in the human head, with each hair strand measuring less than $120\mu m$ in diameter [Rosenblum *et al.*, 1991]. The challenge is even bigger if we need a sequence of images to form an animation, because now we need to emulate the motion of hair, which can be influenced by many things, such as hair length, water and humidity, wind, and collision among the hair strands and other objects. In addition to that, we have to consider the small time budget available to real-time and interactive applications, which is at most $33ms$ per frame [Akenine-Möller *et al.*, 2018].

How realistic should be the motion, or how much artistic freedom should one have when styling a character's hair, depends much on the application for which it's being used. There are also several applications of Computer Graphics outside the entertainment industry that would benefit from enhanced realism that comes from natural motion of hair.

The use of virtual reality is ever more common in education and professional training [Renganayagalu *et al.*, 2021], as well as in therapy and other health-related activities (see [Hoffman *et al.*, 2020], [Smith *et al.*, 2020] and [Emmelkamp and Meyerbröker, 2021]). Many works in that field mention that one of the shortcomings of VR in therapy and education is the poor feeling of social presence and place illusion caused by technical limitations. [Zibrek and McDonnell, 2019] showed evidence that photorealistic characters can improve such feelings. Therefore, techniques that enable simulation of natural hair motion in VR applications can improve the efficacy of education and health tools that use that medium.

Many works have been published in the last three decades about techniques to simulate hair movement and the many ways it can interact with the environment. Those techniques vary in levels of realism, artistic freedom and compute efficiency. Our work will focus on approaches based on Position-Based Dynamics (PBD) [Müller *et al.*, 2007] and will present an improvement to an existing approach, so that hair can keep its curls and hairstyle.

II. Related works

One of the first published works about hair simulation was written more than thirty years ago [Rosenblum *et al.*, 1991]. It described not only how to simulate hair but also how to model and render it. Although the simulation procedures described there are simple and, to a certain degree, inefficient, they laid the foundation to almost all the works that came later, either by people looking for ways to improve it or by people that were looking for alternative approaches.

Unfortunately there seems to be a lack of systematic reviews about hair simulation: besides the brief review of the state of the art present in research papers, the only in-depth review of the topic that we know of was published almost twenty years ago [Ward *et al.*, 2007]. A more recent survey focused on hair modeling [Yang, 2024] cites briefly some works about hair simulation.

We can separate the works on hair simulation in two main groups: the ones that present techniques to simulate individual hair strands and the ones that treat hair collectively. There are also some works that fall in between. We could say that all of them could be placed on a scale where, on one end, we have the finest detail at a high computational cost and, on the other, the coarsest detail with the highest efficiency. This idea is illustrated in Figure 1.

From our studies, we identified that every work followed one of two trends: either advance what "the finest" and

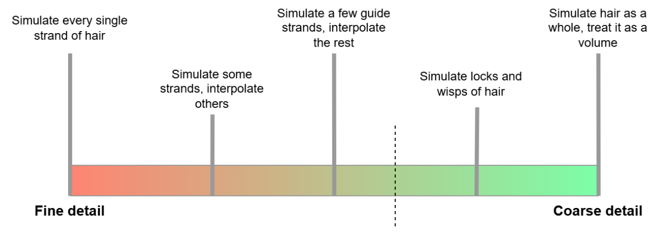

Simulate every single strand of hair
Simulate some strands, interpolate others
Simulate a few guide strands, interpolate the rest
Simulate locks and wisps of hair
Simulate hair as a whole, treat it as a volume
Fine detail
Coarse detail

Fig. 1: Hair simulation techniques in a scale of detail and computational cost

most physical-accurate details are; or optimize performance, realism and artistic freedom. Now we'll dive deeper in each one of the two groups, describing in more details the works presented and trying to place them in the scale above, as well as identifying which trend each work seems to follow. Special attention will be given to PBD-based works afterwards.

A. *Strand-Based Simulation*

[Rosenblum *et al.*, 1991] presented a technique that sought to simulate every strand of hair, placing it in the left-most end of the scale in Figure 1. Their method represents hair strands as mass-spring systems. Although it is simple to implement and it was a pioneer work in the field, it suffers from numerical instability and is limited in regard to which physical phenomena it can simulate. [Selle *et al.*, 2008] pushed the left end of the scale further while still using mass-spring systems. In their work, the authors managed to simulate thousands of hair strands using a new spring layout and a new semi-implicit integration method, allowing them to simulate realistically torsion, bending and inextensibility characteristic of hair strands, however it took the authors several minutes to simulate a single frame. It is interesting to notice that their simulations ran on a multi-core CPU, therefore, porting their implementation to GPU could allow this technique to run in real-time.

Naturally, the next step for mass-spring hair simulations would be to simulate a full head of hair in real-time. [Jiang *et al.*, 2020] achieved this feat by innovating in the integration method. They proposed a framework that produces realistic hair dynamics with self-collisions and other volumetric phenomena. The authors linearized the implicit Euler method to integrate position of the particles and used a Eulerian/Lagrangian hybrid to handle self-interactions. All of that was done in a way that it is efficient to be computed with the parallel architecture of GPUs.

Still in the topic of mass-spring simulations, but now going towards the middle of the scale in Figure 1, we have a work [Chai *et al.*, 2014] that presents a data-driven technique to achieve detailed hair simulation in real-time. In contrast with the works previously described, this one does not attempt to simulate every single strand of hair in real-time. It rather simulates only a few guide strands and then interpolates the rest using a machine learning model. This approach was later improved by [Chai *et al.*, 2016] to handle interpolation of hair strands in scenarios with complex collisions and interactions.

Mass-spring systems are probably the most used representation for hair strands in hair simulations. However, there are other approaches that are worth mentioning. One of such is the work by [Bertails *et al.*, 2006]. It represents hair strands as elastic rods and draws from Cosserat and Kirchhoff theory of rods instead of Hooke's law of elasticity. This technique is able to simulate many types of hair (straight, curly, clumpy, etc.), but is not well suited for real-time simulation. At its time of publication, the way it handled bending and torsion was a novelty, however we can't place this work in the extreme left end of the scale (Fig. 1) due to the fact that the authors interpolated some of the hair strands.

B. *Group-Based Simulation*

One way to achieve simple hair dynamics is to simulate hair strips [Guang and Huang, 2002], which are strips of polygonal mesh or surfaces generated by curves [Liang and Huang, 2003] textured as hair wisps. This approach can be used in real-time applications with little computational cost, since the number of objects simulated is extremely small. However it is limited to straight hair and simple hair motion, and the illusion of a head full of hair falls apart if viewed too close.

Around the same time, another approach was developed. It simulates hair dynamics by simulating wisps in a layered fashion [Plante *et al.*, 2001]: in the first layer, the skeleton curve (similar to a guide strand) moved according to a spring-mass system. This determines the global movement of the hair wisp. Then, in the second layer, a polygonal mesh around the skeleton curve is animated using another spring-mass system, capturing the wisp deformations. Finally, in the third layer, individual hair strands are generated inside the envelope. This approach was further improved by [Plante *et al.*, 2002].

Still on the theme of generating hair strand from the inside of a polygonal mesh, [Yuksel *et al.*, 2009] presented a framework called ”Hair Mesh”, which gives the artist using it great control and creative freedom. This framework allows the artist to model the hairstyle by extruding faces of a polyhedron. After the modeling phase, an algorithm fills the extruded mesh with hair strands. Initially, the authors managed to simulate hair dynamics by treating the vertices of the hair mesh as chains of rigid bodies. Hair mesh simulation was later improved by [Wu and Yuksel, 2016]. In the new version, the authors used sheet-based cloth simulation models and introduced a new volumetric force model to handle collisions better.

There is also an approach that could be considered a hybrid between strand-based and group-based simulation: simulate hair strands individually but at the same time, treat the hair as a continuum fluid. [Hadap and Magnenat-Thalmann, 2001] were one of the first to take that approach. They modeled hair-hair, hair-body and hair-air interactions using smoothed particle hydrodynamics (SPH) and the hair geometry in a strand-by-strand fashion using elastic fiber dynamics.

C. PBD-based Simulation

PBD was first introduced by [Müller *et al.*, 2007] as a simulation framework that is stable, robust and fast to compute, which are characteristics that many times are preferred over accuracy in real-time applications. While many techniques for simulating dynamic objects work with forces or impulses, this approach works with the position of the dynamic bodies. In this framework, the main components that drive the types of dynamics being simulated are the **constraints** on the elements' positions. If we correct the positions so that the constraint is fully satisfied, we call it a **hard constraint**, otherwise it is a **soft constraint**. Unfortunately, PBD doesn't handle soft constraints very well. To remedy that, [Macklin *et al.*, 2016] proposed an improvement to PBD and called it Extended Position-Based Dynamics (XPBD).

Although XPBD is not an exact solver for dynamic objects, the authors demonstrated that the error can be negligible for most applications and that, given enough iterations, it converges closely to the exact result. Some time later, [Macklin *et al.*, 2019] made the discovery that performing sub-steps results in better convergence than iterating over the constraints multiple times in one single time step.

PBD was first used to simulate hair dynamics by [Han and Harada, 2012]. Their approach performed multiple passes of constraint solving each frame so that the particles that constitute the hair strands would converge towards their expected position. The authors used a distance constraint to maintain the hair length, and two shape constraints, one local and one global, to keep the hair form and enable the simulation curls and other hairstyles.

In the same year, [Müller *et al.*, 2012] published a work that presented another approach to simulate hair. Instead of iterating many times over the constraints to converge to a better solution with less length variation of hair strands, the authors proposed a modification to PBD that guarantees inextensibility in only one iteration per time step. They called it **Dynamic Follow-The-Leader** (DFTL).

Some time later, the local shape constraint [Han and Harada, 2012] and DFTL [Müller *et al.*, 2012] were combined into one simulation framework by [Sánchez-Banderas *et al.*, 2015]. They've made two more contributions to PBD-based hair simulation: a method to simulate hair-hair interactions and a data layout to leverage the GPU's memory architecture.

III. New Shape Constraint

Our motivation for the new constraint was to expand the work by [Müller *et al.*, 2012] and add to it the capability of simulating curls and keeping the hairstyle. As we mentioned previously, there is no recent comprehensive survey on the topic of real-time hair simulation, therefore, at the beginning of our research, we still had no knowledge of the works by [Han and Harada, 2012] and [Sánchez-Banderas *et al.*, 2015]. Only when we were near the end of our formulation that we became aware of those works, fortunately, although similar, our technique has enough unique features to be considered a new work of its own. We took inspiration from the data layout from [Sánchez-Banderas *et al.*, 2015] to parallelize our algorithm.

At a high level, before starting the simulation, our algorithm stores the target position of each particle after the root in the local coordinate system of their respective parents. During the simulation, our shape constraint tries to minimize the angle between each particle and its target position in respect to the coordinate system of its predecessor. Conceptually, our approach is very similar to that of [Han and Harada, 2012], however there are some key differences. For instance, while Han and Harada's local shape constraint minimizes the Euclidean distance between particle and its target position, ours minimizes the angle between the particle and the target, passing by the particle's parent.

Another distinction is the method used to compute the local coordinates for each particle. Han and Harada take a more traditional approach by first defining the root's coordinate system and then rotating and translating it for all the next particles. On the other hand, we explored an unorthodox approach that involves projecting an auxiliary point for each particle in the strand. Although our approach is a simplification of the algebraic principles applied by Han and Harada, it brought satisfactory results. Yet another distinction from [Han and Harada, 2012] and [Sánchez-Banderas *et al.*, 2015] is the fact that we used XPBD's formulation for soft constraints.

A. Preparation

We will now describe the process to to compute local coordinate systems and all the other procedures involving our method focusing only on a single strand, therefore we'll index particles in respect with their position in the chain.

We get the local coordinates centered at the root particle in a similar way to [Han and Harada, 2012]: let $\mathbf{p}_0$ be the position of said particle, then we choose an arbitrary point $\mathbf{Aux}_0$ (called auxiliary point) that satisfies the following properties:

1) $|\mathbf{Aux}_0 - \mathbf{p}_0| = 1$
2) $\mathbf{Aux}_0$ is on the plane perpendicular to the unit vector $\mathbf{n}_0$, which is the normal vector of the surface from which the strand is protruding at $\mathbf{p}_0$

With $\mathbf{Aux}_0$ and $\mathbf{n}_0$, we can compute the basis vectors $\mathbf{i}_0$, $\mathbf{j}_0$ and $\mathbf{k}_0$ for the local coordinates relative to $\mathbf{p}_0$:

$$\mathbf{i}_0 = \mathbf{Aux}_0 - \mathbf{p}_0 \tag{1}$$
$$\mathbf{j}_0 = \mathbf{n}_0 \tag{2}$$
$$\mathbf{k}_0 = \mathbf{i}_0 \times \mathbf{j}_0 \tag{3}$$

Let $\mathbf{T}_0$ be the basis matrix from those basis vectors. To simplify our explanation, let's consider that the particles' target positions are their initial positions. Therefore, the target position for the next particle in the chain can be computed by $\mathbf{p}_1^* = \mathbf{T}_0(\mathbf{p}_1 - \mathbf{p}_0)$ (where $\mathbf{p}_1$ is the position of the particle right after the root). We'll need to store the auxiliary point $\mathbf{Aux}_0$, the normal vector $\mathbf{n}_0$ and the target position $\mathbf{p}_1^*$ for the simulation phase.

For the next particles' target positions $\mathbf{p}_i^* = \mathbf{T}_{i-1}(\mathbf{p}_i - \mathbf{p}_{i-1})$, we will need to find the local basis matrix $\mathbf{T}_{i-1}$, relative to $\mathbf{p}_{i-1}$. To do so, we need the basis vectors $\mathbf{i}_{i-1}$, $\mathbf{j}_{i-1}$ and $\mathbf{k}_{i-1}$, which are given by:

$$\mathbf{i}_{i-1} = \mathbf{Aux}_{i-1} - \mathbf{p}_{i-1} \tag{4}$$
$$\mathbf{j}_{i-1} = \frac{\mathbf{p}_{i-1} - \mathbf{p}_{i-2}}{|\mathbf{p}_{i-1} - \mathbf{p}_{i-2}|} \tag{5}$$
$$\mathbf{k}_{i-1} = \mathbf{i}_{i-1} \times \mathbf{j}_{i-1} \tag{6}$$

We compute $\mathbf{Aux}_{i-1}$ by projecting $\mathbf{Aux}_{i-2}$ onto the plane defined by the point $\mathbf{p}_{i-1}$ and the normal vector $\mathbf{p}_{i-1} - \mathbf{p}_{i-2}$, and adjusting it so that $|\mathbf{Aux}_{i-1} - \mathbf{p}_{i-1}| = 1$. This time, we need to store only $\mathbf{p}_i^*$ for the simulation phase.

One might ask why do we do this projection step instead of computing a rotation matrix to find $\mathbf{T}$, like [Han and Harada, 2012] did. Our rationale for that is the fact that projecting the auxiliary point takes less trigonometric operations than computing a rotation matrix.

B. Simulation

In the simulation phase, our algorithm will do something similar to what was done in the preparation phase, except that instead of storing the target position, it will compare the particles' current positions with their respective target position using our formulation for the shape constraint. Our constraint function takes two vectors as parameters, the i-th particle target position and its current position in respect to $i-1$-th local coordinate system.

Our constraint is defined by Eq. 7, with its gradient described by Eq. 8, where $a = \mathbf{v}_1 \cdot \mathbf{v}_2$, $b = |\mathbf{v}_1|^2$, and $c = |\mathbf{v}_2|^2$. We follow XPBD's formulation for soft constraints.

$$C = acos\left(\frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1| \cdot |\mathbf{v}_2|}\right) \tag{7}$$

$$\nabla C = \begin{bmatrix} \dfrac{\mathbf{v}_1.x \cdot a - \mathbf{v}_2.x \cdot b}{b\sqrt{bc - a^2}} \\[2ex] \dfrac{\mathbf{v}_1.y \cdot a - \mathbf{v}_2.y \cdot b}{b\sqrt{bc - a^2}} \\[2ex] \dfrac{\mathbf{v}_1.z \cdot a - \mathbf{v}_2.z \cdot b}{b\sqrt{bc - a^2}} \end{bmatrix} \tag{8}$$

IV. Implementation

We've wrote two implementation for our hair simulation. The first one served as a proof of concept for our constraint formulation, while the later was used to evaluate performance and stress test it.

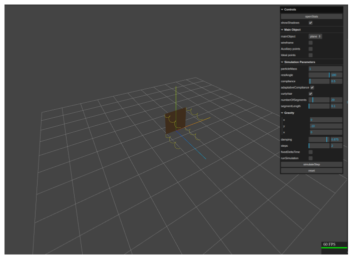
A. Implementation 1: Typescript and WebGL

We chose to use web-based technologies for three main reasons:

1) Modern web frameworks have the feature "hot reload", in which parts of the code are updated without having to interrupt the application, this allows fast iterations during development.
2) High level of abstraction: even when it comes to graphics programming, there are many frameworks and libraries that abstract away much of the boilerplate code and low-level memory management.
3) Having a simulation that runs in a web browser is very useful for live demonstrations.

We've used the library *Three.JS* [Cabello, 2025] to handle rendering and user input. This is a graphics library for the web built on top of WebGL which takes care of low-level buffer management, as well as scene graph management and input processing, allowing users to interact with the virtual 3D space.

ht



As we've mentioned before, our goal for the web implementation was to use it as proof of concept and as a tool to refine our constraint formulation and our algorithm, therefore, performance was not our main concern. A screenshot of our application can be found in the right. It features a parameter menu in which we can change things such as strand length, compliance, damping, etc., as well as enabling debug tools. Besides that, our application featured interactive camera and real-time interactions with the object in the scene.

We've developed a second version of this application, this time with predefined parameters and simplified user interface optimized for mobile devices. It was used as a live demo for a presentation done at *2024 ACM SIGGRAPH*

Symposium on Interactive 3D Graphics and Games (I3D) [Cavazotti and Guedes, 2024].

B. Implementation 2: Python and CUDA

The goal of our second implementation was to run the simulation on a GPU. After exploring many libraries and APIs, we decided to use *Taichi*, "an open-source library [...] which alleviates this practical issue by providing an accessible, portable, extensible, and high-performance infrastructure that is reusable and tailored for computer graphic" [Hu, 2018]. This library is a "domain-specific language embedded in Python" [Hu, 2025] that JIT compiles Python-like code into machine code for the CPU or GPU, depending on the client's hardware. In our case, our simulation was compiled into CUDA because we have NVIDIA GPUs.

In our second implementation, we focused on performance and tooling rather than interactivity, therefore, users can't orbit the camera nor move the object around. On the other hand, we've implemented mechanisms to run automated tests, gather data and generate reports.

Regarding parallelization, we assigned one hair strand for each thread, which iterates over all the particles that constitutes the strand, updating their state and applying constraints. We could have implemented a more sophisticate parallelization approach to reduce serial computation, like the one used by [Han and Harada, 2012], however that would require more sub-steps and defeats the purpose of DFTL, which is to simulate hair dynamics with only one sub-step per frame. That being said, our parallelization strategy was similar to the one employed [Sánchez-Banderas *et al.*, 2015].

We've implemented three constraints in our application: the new shape constraint, a distance constraint like the one used in [Müller *et al.*, 2012] and a simple penetration constraint that moves the particles out of the spherical colliders.

V. Results

We've performed most of the qualitative tests using the web implementation. Quantitative tests regarding performance and error margin were done in the GPU implementation. While the web version was tested on many devices, the GPU version was tested on a *Acer Predator Helios 300* notebook with 16 *GB* of RAM, a 9th generation *Intel Core i7* processor and a *NVIDIA GeForce RTX 2060* GPU.

A. Quantitative tests

We compared two levels of optimization in data layout with a naive implementation. In the naive version, we used a "struct-of-arrays" approach, in other words, we had an array for particle positions, another for particle velocities, and another for target positions, and so on, as shown in Figure 2. For those familiar with PBD, it's clear that we have an inefficient memory access pattern with this implementation.

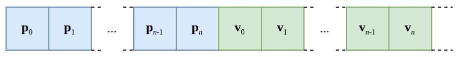


Fig. 2: Naive data layout

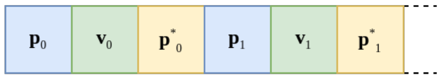


Fig. 3: Array-of-structs

This leads us to the first level of optimization: "array-of-structures". In this layout, we group together all the data related to each particle, as represented in Figure 3. This way, the GPU doesn't need to load different blocks of data to simulate one particle. However, both the naive layout and this one have a problem in common: two neighboring threads are not processing data that live in neighboring memory addresses. That causes increased memory bandwidth usage and lower throughput, since threads from the same group might be working with data from different pages.

This issue was addressed by [Sánchez-Banderas *et al.*, 2015] in their "batch" data layout. Instead of having the data of all the particles of a strand in contiguous spaces of memory (see Figure 4), we can allocate the first particles of every strand together, then the second particles, and so on, as illustrated in Figure 5. In this layout, it's less likely that threads will be blocked because of memory access, and although the stride done on memory addresses in the kernel's main loop is bigger and may cause new pages of memory to be loaded, it will not block any threads.

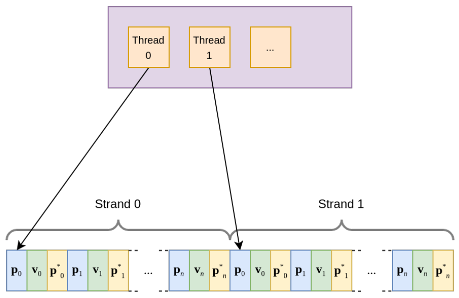


Fig. 4: Schema of memory access in "array-of-structs" layout

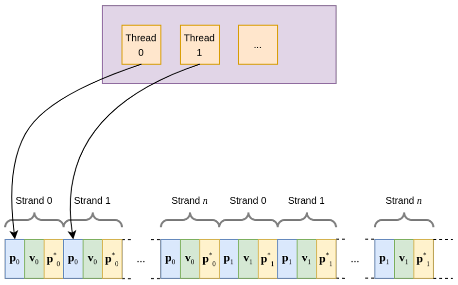
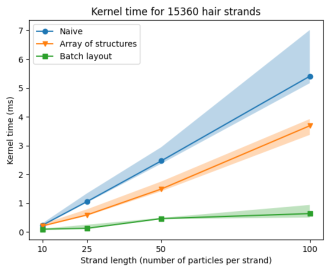
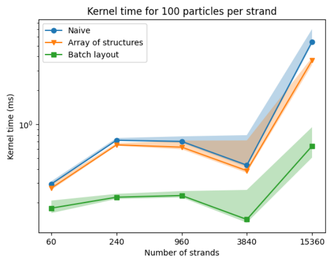

Thread 0
Thread 1
...
Strand 0
Strand 1
Strand n
Strand 0
Strand 1
Strand n
$\mathbf{p}_0$ $\mathbf{v}_0$ $\mathbf{p}^*_0$ $\mathbf{p}_0$ $\mathbf{v}_0$ $\mathbf{p}^*_0$ ...
$\mathbf{p}_0$ $\mathbf{v}_0$ $\mathbf{p}^*_0$ $\mathbf{p}_1$ $\mathbf{v}_1$ $\mathbf{p}^*_1$ $\mathbf{p}_1$ $\mathbf{v}_1$ $\mathbf{p}^*_1$ ...
$\mathbf{p}_1$ $\mathbf{v}_1$ $\mathbf{p}^*_1$

Fig. 5: Schema of memory access in "batch" layout

We've tested our implementation with each one of the three data layouts, and for each version, we've experimented with varying number of hair strands and particles per strand. Figure 6a, which depicts kernel execution times with the heaviest workloads we tested:



(a) Kernel time for 15360 hair strands



(b) Kernel time for 100 particles per strand

Fig. 6

In this figure, the colored areas represent the maximum and the minimum time recorded throughout the 500 frames simulated, and the lines represent the average. The number of particles simulated ranged from around 153 thousand to more than 1.5 million. In the chart we can see how much the "batch" data layout improved performance, with average kernel execution time for 100 particles per strand dropping from 5.4 *ms* in the naive layout to only 0.64 *ms* in the optimized version.

Since we have parallelism across strands, and particles from the same strand are updated sequentially, we could expect that the kernel execution time for varying number of strands and fixed number of particles per strand would remain constant. However that is not what we've observed (see Fig. 6b).

The increase of execution time observed when comparing 60 and 240 strands is due to memory access pattern: the less optimized versions had to load more pages of memory, and that caused the significant increase in execution time. Then, the time is reduced at 960 and 3840, probably indicating that the data was more aligned with memory page size. The spike in execution time at 15360 is probably also caused by inefficient memory access patterns, but we couldn't pinpoint the specific cause, and our knowledge only goes so far to make assumptions.

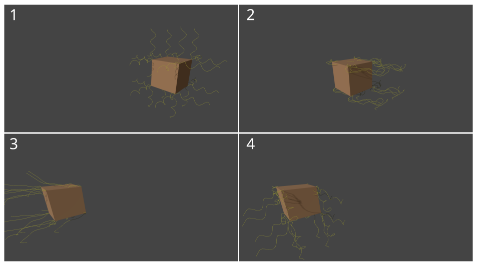
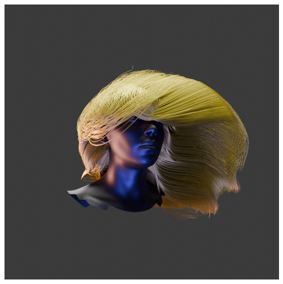


Fig. 7: Recovery of curls after deformation

B. *Qualitative test*



Now, regarding the visual quality of our algorithm, we can see in inset figure the rendered result of a simulation with more than 17 thousand hair strands and more than 1.3 million particles. We've observed that our constraint can emulate appearance of volumetric hair and that it can recover its shape even after violent motions. Our constraint can also simulate curly hair, as seen in the minimal demo shown in Figure 7. Each image is a few frames apart from each other. As we can see, the curls can recover their shape even after undergone severe deformation, albeit sagging.

We would like to compare our results both in terms of performance and visual quality with the works that closely related to ours ([Müller *et al.*, 2012], [Han and Harada, 2012] and [Sánchez-Banderas *et al.*, 2015]), but we couldn't find their implementations, making it impossible to do any type of comparison.

C. *Public Reception*

In May 2024, we presented our work at *2024 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. Even if it was not finished at the time, our work was received with interest by the research community, being awarded the prize for "Best Poster" by community vote [Cavazotti and Guedes, 2024].

VI. Conclusion

We have reached our goal to expand the technique developed by [Müller *et al.*, 2012], enabling it to simulate curly hair and other hair styles. Although the simulation quality of our algorithm is not on par with some other techniques presented in article, it received interest from the community, showing its potential. In terms of performance, we are very pleased, since we've managed to simulate more than 1.5 million particles in a time budget compatible VR applications.

Declarations

Funding

This project was partially supported by CNPq (420079/2021-1) and PROEX CAPES.

Materials

The source code and the data we've produced are available at: https://gitlab.c3sl.ufpr.br/teoria/hair-simulation.

References

[Akenine-Möller *et al.*, 2018] Akenine-Möller, T., Haines, E., and Hoffman, N. (2018). *Real-time rendering*. CRC Press.

[Bertails *et al.*, 2006] Bertails, F., Audoly, B., Cani, M.-P., Querleux, B., Leroy, F., and Lévêque, J.-L. (2006). Super-helices for predicting the dynamics of natural hair. *ACM Transactions on Graphics (TOG)*, 25(3):1180–1187.

[Cabello, 2025] Cabello, R. (2010–2025). Three.js. https://threejs.org/. Accessed: 2025-04-21.

[Cavazotti and Guedes, 2024] Cavazotti, M. d. M. and Guedes, A. L. P. (2024). Real-time curls. https://i3dsymposium.org/2024/posters.html. SIGGRAPH - I3D - Awarded Best poster - audience choice - https://i3dsymposium.org/2024/awards.html#best-poster---audience-choice.

[Chai *et al.*, 2014] Chai, M., Zheng, C., and Zhou, K. (2014). A reduced model for interactive hairs. *ACM Transactions on Graphics (TOG)*, 33(4):1–11.

[Chai *et al.*, 2016] Chai, M., Zheng, C., and Zhou, K. (2016). Adaptive skinning for interactive hair-solid simulation. *IEEE transactions on visualization and computer graphics*, 23(7):1725–1738.

[Emmelkamp and Meyerbröker, 2021] Emmelkamp, P. M. and Meyerbröker, K. (2021). Virtual reality therapy in mental health. *Annual review of clinical psychology*, 17(1):495–519.

[Guang and Huang, 2002] Guang, Y. and Huang, Z. (2002). A method of human short hair modeling and real time animation. *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*. DOI: 10.1109/pccga.2002.1167891.

[Hadap and Magnenat-Thalmann, 2001] Hadap, S. and Magnenat-Thalmann, N. (2001). Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338. DOI: 10.1111/1467-8659.00525.

[Han and Harada, 2012] Han, D. and Harada, T. (2012). Real-time hair simulation with efficient hair style preservation. In Bender, J., Kuijper, A., Fellner, D. W., and Guerin, E., editors, *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association. DOI: /10.2312/PE/vriphys/vriphys12/045-051.

[Hoffman *et al.*, 2020] Hoffman, H. G., Boe, D. A., Rombokas, E., Khadra, C., LeMay, S., Meyer, W. J., Patterson, S., Ballesteros, A., and Pitt, S. W. (2020). Virtual reality hand therapy: A new tool for nonopioid analgesia for acute procedural pain, hand rehabilitation, and vr embodiment therapy for phantom limb pain. *Journal of hand therapy*, 33(2):254–262.

[Hu, 2018] Hu, Y. (2018). Taichi: An open-source computer graphics library. *arXiv preprint arXiv:1804.09293*.

[Hu, 2025] Hu, Y. (2018–2025). Taichi. https://taichi.graphics/. Accessed: 2025-04-22.

[Jiang *et al.*, 2020] Jiang, J., Sheng, B., Li, P., Ma, L., Tong, X., and Wu, E. (2020). Real-time hair simulation with heptadiagonal decomposition on mass spring system. *Graphical Models*, 111:101077.

[Liang and Huang, 2003] Liang, W. and Huang, Z. (2003). An enhanced framework for real-time hair animation. *11th Pacific Conference on Computer Graphics and Applications, 2003. Proceedings.*. DOI: 10.1109/pccga.2003.1238296.

[Macklin *et al.*, 2016] Macklin, M., Müller, M., and Chentanez, N. (2016). Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, pages 49–54.

[Macklin *et al.*, 2019] Macklin, M., Storey, K., Lu, M., Terdiman, P., Chentanez, N., Jeschke, S., and Müller, M. (2019). Small steps in physics simulation. In *Proceedings of the 18th annual ACM siggraph/eurographics symposium on computer animation*, pages 1–7.

[Müller *et al.*, 2007] Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118.

[Müller *et al.*, 2012] Müller, M., Kim, T.-Y., and Chentanez, N. (2012). Fast simulation of inextensible hair and fur. *VRIPHYS*, 12:39–44.

[Plante *et al.*, 2001] Plante, E., Cani, M.-P., and Poulin, P. (2001). A layered wisp model for simulating interactions inside long hair. *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*. DOI: 10.1007/978-3-7091-6240-8_13.

[Plante *et al.*, 2002] Plante, E., Cani, M.-P., and Poulin, P. (2002). Capturing the complexity of hair motion. *Graphical Models*, 64:40–58. DOI: 10.1006/gmod.2002.0568.

[Renganayagalu *et al.*, 2021] Renganayagalu, S. K., Mallam, S. C., and Nazir, S. (2021). Effectiveness of vr head mounted displays in professional training: A systematic review. *Technology, Knowledge and Learning*, pages 1–43.

[Rosenblum *et al.*, 1991] Rosenblum, R. E., Carlson, W. E., and Tripp, E. (1991). Simulating the structure and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation*, 2(4):141–148. DOI: 10.1002/vis.4340020410.

[Sánchez-Banderas *et al.*, 2015] Sánchez-Banderas, R. M., Barreiro, H., García-Fernández, I., and Pérez, M. (2015). Real-time inextensible hair with volume and shape. In *CEIG*. DOI: 10.2312/ceig.20151194.

[Selle *et al.*, 2008] Selle, A., Lentine, M., and Fedkiw, R. (2008). A mass spring model for hair simulation. In *ACM SIGGRAPH 2008 papers*, pages 1–11.

[Smith *et al.*, 2020] Smith, V., Warty, R. R., Sursas, J. A., Payne, O., Nair, A., Krishnan, S., da Silva Costa, F., Wallace, E. M., Vollenhoven, B., *et al.* (2020). The effectiveness of virtual reality in managing acute pain and anxiety for medical inpatients: systematic review. *Journal of medical Internet research*, 22(11):e17980.

[Ward *et al.*, 2007] Ward, K., Bertails, F., Kim, T.-y., Marschner, S. R., Cani, M.-p., and Lin, M. C. (2007). A Survey on Hair Modeling: Styling, Simulation, and Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):213–234. DOI: 10.1109/TVCG.2007.30.

[Wu and Yuksel, 2016] Wu, K. and Yuksel, C. (2016). Real-time hair mesh simulation. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 59–64.

[Yang, 2024] Yang, J. (2024). A survey on hair modeling. *Highlights in Science, Engineering and Technology*, 115:512–526. DOI: 10.54097/y7bzrg65.

[Yuksel *et al.*, 2009] Yuksel, C., Schaefer, S., and Keyser, J. (2009). Hair meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)*, 28(5):166:1–166:7. DOI: 10.1145/1661412.1618512.

[Zibrek and McDonnell, 2019] Zibrek, K. and McDonnell, R. (2019). Social presence and place illusion are affected by photorealism in embodied VR. In *Proceedings of the 12th ACM SIGGRAPH Conference onf Motion, Interaction and Games*, pages 1–7.