Simulation of Inextensible Hair and Fur

Tom Möbert & Mathias Putz

2. February 2017

seit 1558

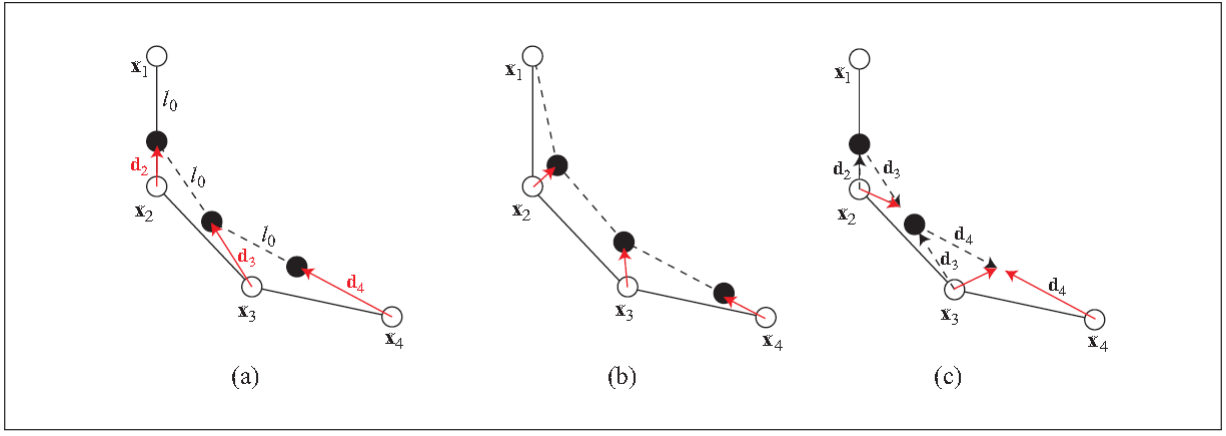
- We focus on the fast simulation of hair and fur on animated characters.
- **Where is it used?** Films, computer games or related
- **Whats the difficulty?** The sheer number of hair strands and the fact that each hair is inextensible. Keeping thousands of deformable objects from being stretched is computationally expensive.
- To do this in **real time** and **naturally**.

The following algorithm is described in:
M. Müller, T.Y. Kim, N. Chentanez, Fast Simulation of Inextensible Hair and Fur, Darmstadt, Germany, December 6-7 2012.

- We model a single hair by a chain of particles that is attached at one end $(x_1, ..., x_n)$.
- All particles have the same distance l_0 in between.
- We start at a particle that violate the distance constraints, commonly x_1.
- We iterate through all the particles once and move the particles such that all the constraints are satisfied.

Process called Follow-The-Leader (a).

- Process the particles in the order from 2 to n.
- Particle i has to be on a sphere with radius l_0 around particle $i-1$.
- Choose the point on the sphere that is closest to the original position x_i.



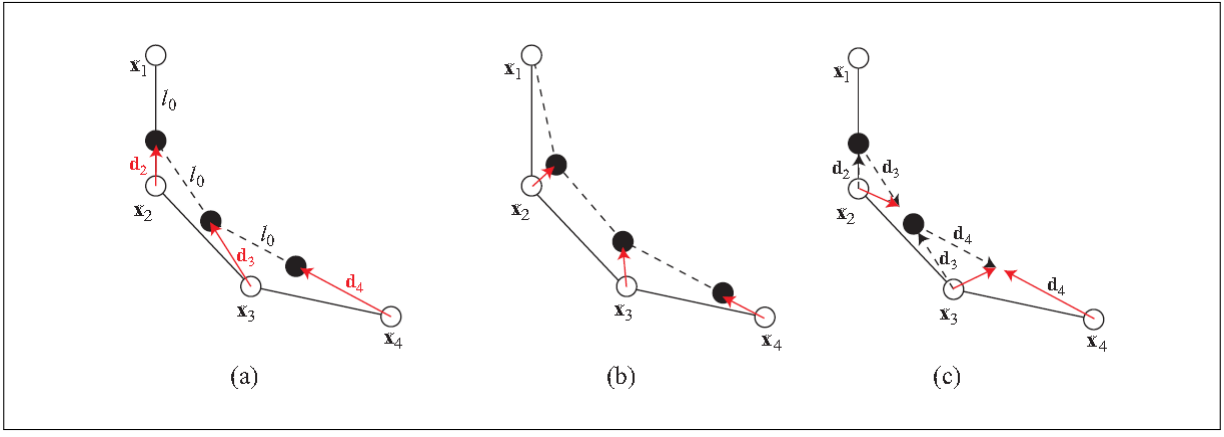
- Need to store and update velocities $v_1, ..., v_n$ along with the positions.
- A way to derive the velocity updates from the position updates is called **Position Based Dynamics (PBD)** (b).

$$x + \Delta t v + \Delta t^2 f \rightarrow p \tag{1}$$

$$solve\_constraints(p) \rightarrow p \tag{2}$$

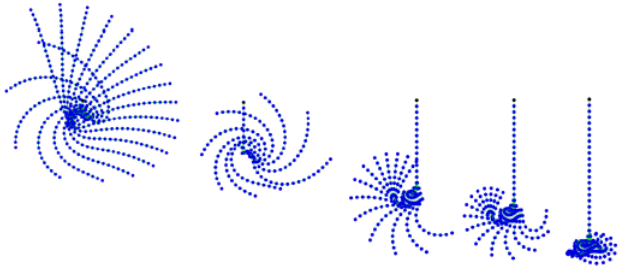
$$\frac{x - p}{\Delta t} \rightarrow v \tag{3}$$

$$p \rightarrow x \tag{4}$$

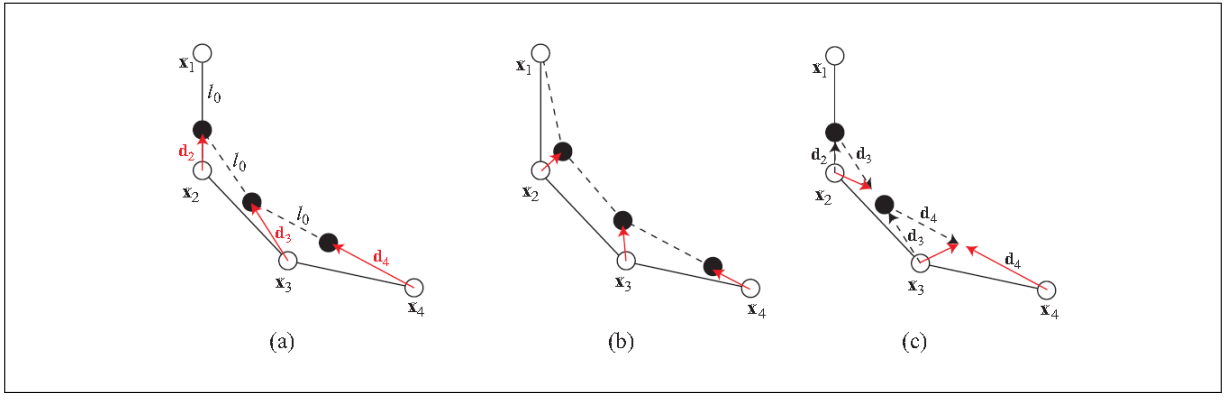


Without respect to the mass of the particles, the method above ends up in a strange behavior.
To fix that bug we need a **velocity correction** by using damping (c).

$$\frac{x_i - p_i}{\Delta t} + s_{damping}\frac{-d_{i+1}}{\Delta t} \rightarrow v_i \qquad (5)$$



With the correction vector d_i and $s_{damping} \in [0, 1]$.

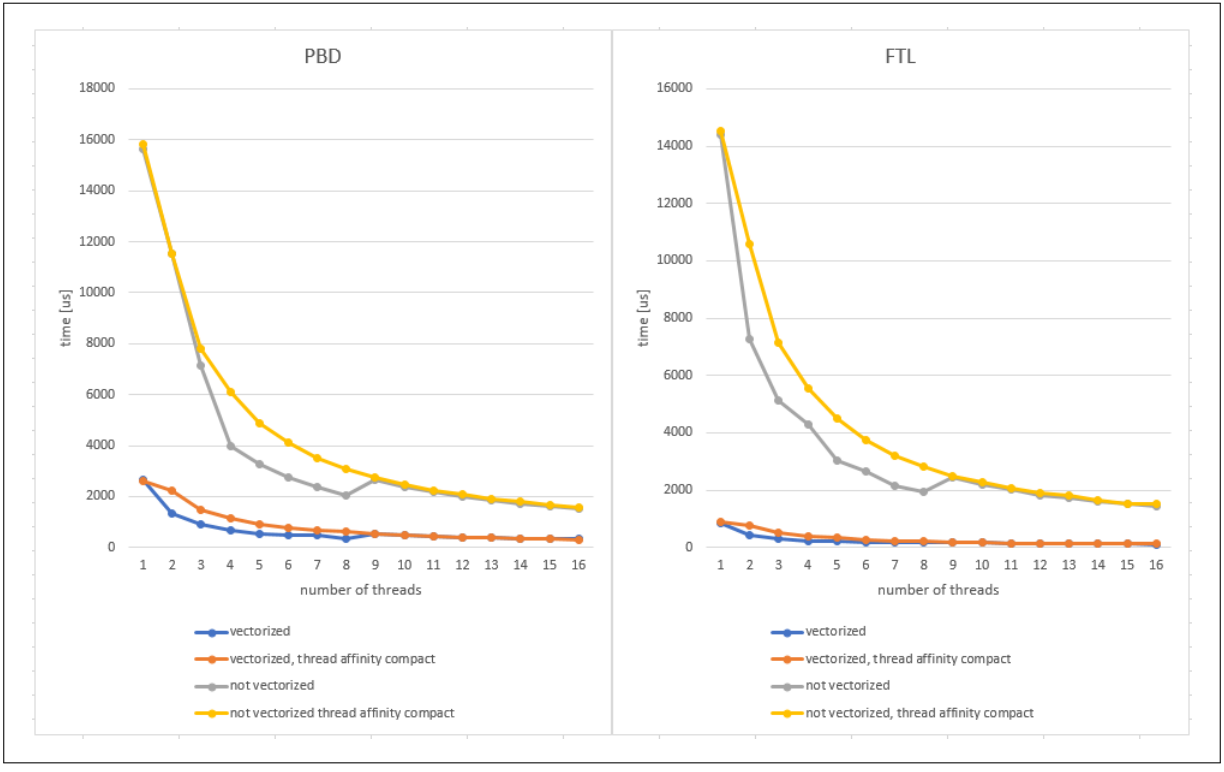


Our implementation

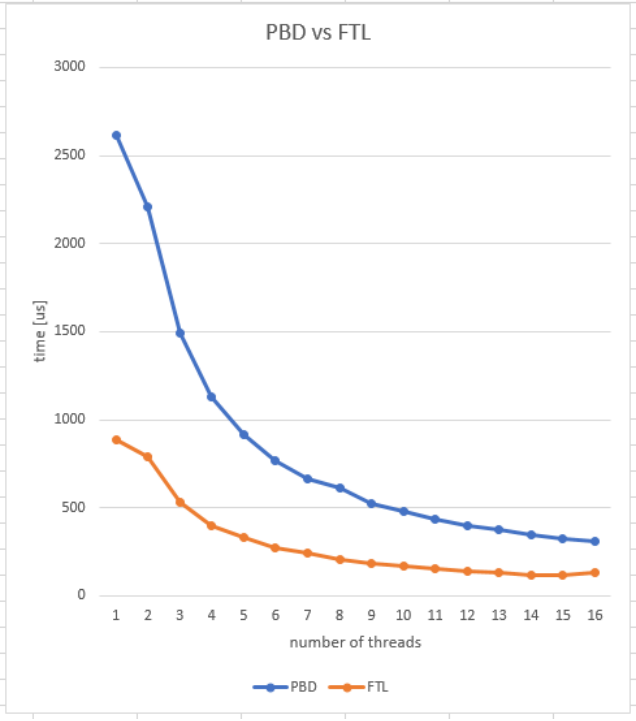
- openGL and openMP is used
- implementation of 2 different algorithms
- display and save simulation
- used aligned data-structure (64 byte without padding) for all vertexes
- debug vs release build speedup: 6 @ PBD, 16 @ FTL
- measured average time of 10000 pictures with 1000 strands and 50 vertexes each
- speedup:

threads	2	4	8	12	16
FTL	1.4	2.6	5.1	7.6	9.7
PBD	1.4	2.6	5.2	7.6	10.0

Times


PBD
time [us]
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
number of threads
vectorized
vectorized, thread affinity compact
not vectorized
not vectorized thread affinity compact
FTL
time [us]
16000
14000
12000
10000
8000
6000
4000
2000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
number of threads
vectorized
vectorized, thread affinity compact
not vectorized
not vectorized, thread affinity compact

Times


PBD vs FTL
3000
2500
2000
1500
1000
500
0
time [us]
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
number of threads
PBD
FTL