Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016)
Ladislav Kavan and Chris Wojtan (Editors)

Position and Orientation Based Cosserat Rods

T. Kugelstadt and E. Schömer

Johannes Gutenberg University Mainz, Germany

Abstract

We present a novel method to simulate bending and torsion of elastic rods within the position-based dynamics (PBD) framework. The main challenge is that torsion effects of Cosserat rods are described in terms of material frames which are attached to the centerline of the rod. But frames or orientations do not fit into the classical position-based dynamics formulation. To solve this problem we introduce new types of constraints to couple orientations which are represented by unit quaternions. For constraint projection quaternions are treated in the exact same way as positions. Unit length is enforced with an additional constraint. This allows us to use the strain measures form Cosserat theory directly as constraints in PBD. It leads to very simple algebraic expressions for the correction displacements which only contain quaternion products and additions. Our results show that our method is very robust and accurately produces the complex bending and torsion effects of rods. Due to its simplicity our method is very efficient and more than one order of magnitude faster than existing position-based rod simulation methods. It even achieves the same performance as position-based simulations without torsion effects.

Categories and Subject Descriptors (according to ACM CCS): I.6.8 [Computer Graphics]: Simulation and modeling—Animation

1. Introduction

The simulation of elastic rods has been an active research topic in computer graphics for more than a decade. Its applications reach from simulation of sutures and catheters in virtual surgery over the simulation of ropes, cables, hoses and knots to the simulation of realistically moving vegetation and hair or fur for virtual characters. Due to the fast development of massively parallel graphics hardware and GPGPU, simulation of hair and fur in real-time applications has become very popular recently. Examples from the games industry are NVIDIA Hairworks and AMD TressFX which can simulate ten-thousands of hair strands in real-time. These techniques model strands as particles which are connected by distance constraints or springs. This allows fast and stable simulations. But one drawback is that twisting effects and rod configurations with initial twist such as curls or helices cannot be represented by pure particle systems (at least not without tricks like ghost particles [USS14] or additional springs [SLF08]).

In contrast to these methods there are models based on Cosserat theory. There elastic rods are modeled as a continuous one dimensional curve in 3d-space. An orthonormal frame is attached to every point of the curve and moves along with it (see fig 2). This orientation information can be used to define deformation energy densities for the stretch, shear, bend and twist degrees of freedom. Applying Lagrangian field theory leads to a system of non-linear PDEs which can be solved using finite elements (FEM) or finite difference (FDM) methods. This results in a physically accurate simulation of elastic rods which shows non-linear effects like out-of-plane buckling [LLA11]. The main drawback of Cosserat models is that simulations of rods with high stretching resistance involves solving stiff differential equations. They require very small time steps when integrated with simple explicit schemes or solving large systems of linear equations when integrated implicitly.

It is a promising idea to combine the particle models and Cosserat models. Recently Umetani, Schmidt and Stam [USS14] presented the position-based elastic rods model. They combined the Cosserat model with position-based dynamics (PBD) [MHHR07]. Their model includes torsion and non-linear effects, is unconditionally stable and computationally cheap enough for interactive applications. It is implemented in the Nucleus visual effects engine of Autodesk Maya which underlines the practical relevance of this technique. In PBD all objects are modeled as particles which are coupled by constraint functions. Because frames cannot be handled directly in PBD, the authors fit the Cosserat model into the PBD framework by representing frames as ghost particles. Constraints are used to force the ghost particles to move along with the curve during the simulation process. Bending and twisting resistance is modeled as constraints between the particles and ghost particles.

Our contribution. Following the ideas of Umetani et al. we propose an alternative approach to combine the Cosserat rod model with PBD. Our approach is similar to [USS14] in many aspects but goes in the opposite direction. We enhance the PBD framework so that it becomes possible to define and solve constraints between orthonormal frames directly. To achieve this, we represent the frames as unit quaternions and constraints are defined as

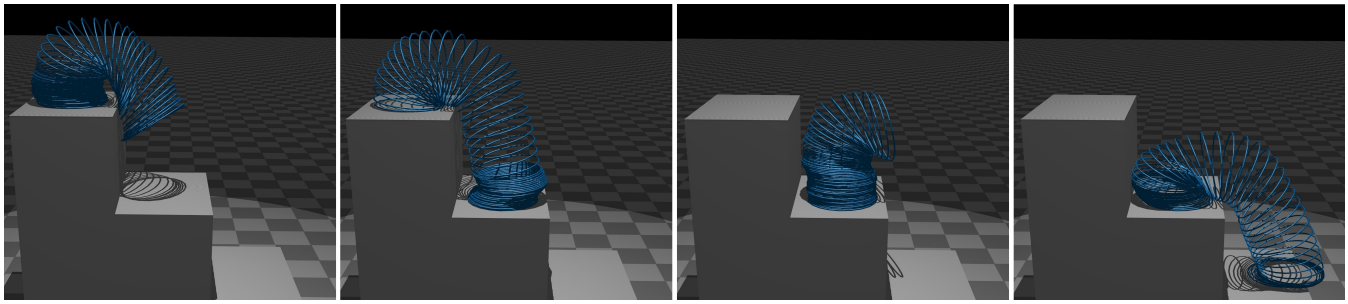

Figure 1: *Slinky walking down a stairway. It has 50 curls and is modeled with 1000 discrete rod elements. It was simulated using 50 solver iterations and took 7ms per frame (without collision detection) on a single core of an Intel Core i5 CPU.*

functions taking particle positions and quaternions as arguments. During the iterative constraint projection process quaternions are treated in the exact same way as positions. They only require an additional constraint to ensure unit length, because only unit quaternions represent proper rotations or frames. This enables us to directly minimize the strain measures from Cosserat theory with the position-based solver. Further we can simply work with the staggered grid discretization which is commonly used in Cosserat rod simulation [GS07, ST07, LLA11] and leads to intuitive geometric interpretations of our constraints. In order to compute the correction displacements of particles and quaternions, gradients of quaternion functions are needed. We show that they can be found with little effort by using a matrix-vector product representation of the quaternion product. The resulting displacement formulas are simple algebraic expressions which can be implemented easily and are very cheap to compute. Due to the fact that our constraints couple relative positions and relative orientations we can also use the rod constraints to couple the rods endpoints with other systems. For example they can be attached to triangles, rigid bodies or coupled with other particle systems.
Our results show that the proposed method is able to produce the same visual quality and has the same robustness as position-based elastic rods, but it needs one order of magnitude less computation time. Another comparison to the simulation of strands in PBD, which are only modeled with distance and bending constraints, shows the high performance of our approach. Even in this case our method is slightly faster. As a challenging benchmark we chose to simulate a Slinky toy which travels down the stairs (see figure 1). In this scene torsion effects, anisotropic bending stiffness and robust collision handling are essential. Because of its simplicity, flexibility, high efficiency and unconditional stability our approach is well suited for a broad spectrum of applications in real-time physics and animation.

Organization. In the following section we discuss related work in rod simulation and position-based dynamics. To keep our paper self-contained we recap the main concepts of quaternions, which are the key component of our technique, in section 3. Thereby we focus on rotations and the derivatives of quaternion functions which are needed to solve the constraints. Section 4 summarizes the main ideas of PBD, including vector constraints [USS14] and shows how orientation quaternions are introduced in the algorithm. In section 5 we introduce the main concepts of continuous Cosserat rods and show how they are discretized. This leads to rod constraints which are presented in section 6. The resulting correction displacements are derived in full detail. In section 7 we discuss implementation details and in section 8 we show results of our method. We compare it to position-based elastic rods and to standard PBD strands with distance and bending constraints. The final section draws a conclusion and discusses possible future work.

2. Related work

The most closely related work is the position-based elastic rods model by Umetani et al. [USS14] which was already discussed in the introduction. Further there is a huge amount of related work in rod simulation, hair simulation and position-based dynamics.

Implicitly discretized Cosserat rods: The Cosserat rod model was introduced to the computer graphics community by Pai [Pai02] who used it to simulate sutures and catheters in virtual surgery. He simulated unshearable and inextensible rods by using curvature as minimal coordinates. It results in an efficient and physically accurate simulation. But the centerline is only represented implicitly by the rod's curvature and has to be recovered by integrating the curvature from one end to the other. This makes collision handling difficult. Further his model does not handle dynamics. This approach was extended by Bertails et al. [BAC*06] to the super-helix model, which was designed to simulate the dynamics of helical rods like curly hair. This method was further improved by Bertails [Ber09] so that time complexity is linear in the number of discrete rod elements. Due to the fact that these models eliminate all constraints by using minimal coordinates, it is not clear how to combine them with PBD.

Explicitly discretized Cosserat rods: Many alternative approaches use an explicit representation of the centerline. Grégoire and Schömer [GS07] modeled cables as connected linear elements which are described by a position and an orientation. The orientations are represented as unit quaternions. They derive constraint energies and the corresponding forces. Therewith they can find static equilibria of cables in a virtual assembly simulation.
This approach was extended to simulate the dynamics of rods by Spillmann and Teschner [ST07]. For their CORDE model they de-

rive discrete energies by applying the finite element method to continuous deformation energies from Cosserat theory. The equations of motion for the discrete rod are found with the Euler-Lagrange equations and solved with an explicit Euler scheme. Because of the explicit representation of the centerline, collisions can be handled efficiently, which allows them to simulate looping effects and knots at interactive rates [SBT07, ST08]. But the explicit time integration requires very small time steps and strong damping to remain stable, which is the main performance bottleneck of their simulation.
A similar approach was presented in the mathematical engineering community by Lang et al. [LL09, LLA11]. They use the same geometric model for the discrete rod. But they derive continuous equations of motion (PDEs) by applying Lagrangian field theory. These are solved by using the finite difference method and standard solvers for stiff differential equations. It results in highly accurate simulations of elastic rods which are crucial in engineering applications. On the contrary our approach aims for simplicity, visually pleasing results and high computational performance. But their work provides an excellent theoretical background of Cosserat rods and how to use quaternions in this context.
Due to the fact that centerline, elastic energy and constraints are modeled explicitly, these models are ideal to be combined with PBD. Basically our approach can be interpreted as a position-based version of the force-based CORDE model. Thereby we can reach unconditional numerical stability, eliminate time step requirements and achieve high computational performance.

Other models: A lot of other rod models have been proposed by the computer graphics community. Bergou et al. [BWR*08] proposed a model based on discrete differential geometry, which was later extended to simulate discrete viscous threads [BAV*10]. Also mass-spring models [BW98, CCK05, IMP*13] are very popular because they are simple, well-investigated, computationally cheap and deliver visually plausible results. But torsion effects cannot be modeled by a serial chain of mass points which are connected by springs. They can be modeled using additional springs [SLF08] which goes along with additional computational costs. A further problem with mass-spring systems is that material properties are difficult to tune because the only material parameter is the stiffness constant of the springs. Recently Michels et al. [MMS15] addressed these issues by using cuboidal oscillator networks and an exponential time integrator. Another approach is to model rods as a serial chain of rigid bodies which are connected by joints [Had06]. It delivers good results but is too expensive for real-time applications. A further method that is closely related to PBD is the tridiagonal matrix algorithm by Han and Harada [HH13]. It is used in AMD's TressFX for real-time simulation of hair in games. Single hair strands are modeled as a serial chain of mass points which are coupled by distance and bending constraints. Due to the simple chain topology all constraints can be satisfied in one step by solving a tridiagonal linear system. This algorithm can be easily parallelized on modern GPUs and is fast enough to simulate a head full of hair in real time, but it does not handle torsion effects.

Position-based dynamics: PBD is a simulation framework which is well suited for games and computer animation because it is fast, robust, versatile and easy to implement and parallelize. It is not accurate enough to reproduce real-world experimental results but it produces visually plausible effects at low computational costs. PBD is implemented in many physics engines e.g. NVIDIA PhysX, Havok Cloth, Bullet and in the Nucleus visual effects engine of Autodesk Maya [Sta09].
Since the first publication by Müller et al. [MHHR07] where PBD was used to simulate cloth and soft bodies, there have been various extensions. The simulation of stiff cloth can be sped up with a hierarchical solver [Mue08] or long range attachment constraints [KCMF12]. Further PBD can be used to simulate inextensible hair/fur [MCK12], real-time fluids [MM13], continuous materials [BKCW14], rigid bodies [DCB14] and elastic rods [USS14]. A survey on the latest developments in the field of PBD can be found in [BMOT13]. Due to this wide range of applications PBD can be used to build a unified physics engine. Recently Macklin et al. presented two-way coupling of fluids, cloth, rigid bodies, deformable bodies and gases [MMCK14], which is implemented in NVIDIA FleX.
A position-based method which is very closely related to our approach is strain-based dynamics [MCKM14]. They use the PBD solver to minimize strain of 2d and 3d deformable models. We also use the PBD solver to minimize strain of the Cosserat rod model. Another deformable solid animation method which is closely related to PBD is shape matching [MHTG05]. It can be also used to simulate hair strands [RKN10]. Oriented particles [MC11] are a further generalization of shape matching. They are related to our work because they also use orientation quaternions in PBD. But they do not couple the oriented particles with orientation constraints. The couplings are achieved through shape matching and the orientations are used to guarantee numerical stability of polar decompositions.

3. Quaternions

Quaternions will be denoted as 4d vectors $q = (q_0, q_1, q_2, q_3)^T = (q_0, \boldsymbol{q}^T)^T \in \mathbb{H}$ with the scalar part q_0 and the vector part $\boldsymbol{q}$. The scalar or real part will be also denoted as $q_0 = \Re(q)$ and the vector or imaginary part as $\boldsymbol{q} = \Im(q)$. The product of two quaternions p and q is defined as

$$pq = \begin{pmatrix} p_0 q_0 - \boldsymbol{p}^T \boldsymbol{q} \\ p_0 \boldsymbol{q} + q_0 \boldsymbol{p} + \boldsymbol{p} \times \boldsymbol{q} \end{pmatrix}. \tag{1}$$

It will be useful to express this product as the matrix-vector product

$$pq = \mathcal{Q}(p)q = \begin{pmatrix} p_0 & -\boldsymbol{p}^T \\ \boldsymbol{p} & p_0 \mathbf{1}_{3\times 3} + [\boldsymbol{p}]^\times \end{pmatrix} \begin{pmatrix} q_0 \\ \boldsymbol{q} \end{pmatrix}. \tag{2}$$

The skew symmetric matrix $[\boldsymbol{p}]^\times$ is used to write the cross product $\boldsymbol{p} \times \boldsymbol{q} = [\boldsymbol{p}]^\times \boldsymbol{q}$ as a matrix-vector product. The matrix $\mathcal{Q}(p)$ is called quaternion matrix. It is also possible to rewrite the product so that the right quaternion is represented as a matrix

$$pq = \hat{\mathcal{Q}}(q)p = \begin{pmatrix} q_0 & -\boldsymbol{q}^T \\ \boldsymbol{q} & q_0 \mathbf{1}_{3\times 3} - [\boldsymbol{q}]^\times \end{pmatrix} \begin{pmatrix} p_0 \\ \boldsymbol{p} \end{pmatrix}. \tag{3}$$

Products with the conjugate quaternion $\bar{q} = (q, -\boldsymbol{q}^T)^T$ can be written using the transposed quaternion matrices

$$\bar{p}q = \mathcal{Q}(p)^T q, \qquad p\bar{q} = \hat{\mathcal{Q}}(q)^T p. \tag{4}$$

Further useful properties of the quaternion matrix are $\mathcal{Q}(p) + \mathcal{Q}(q) = \mathcal{Q}(p+q)$ and $\mathcal{Q}(p)\mathcal{Q}(q) = \mathcal{Q}(pq)$.

Rotations: Later on we will use quaternions as an elegant way of describing rotations of 3d vectors. A unit quaternion q with length $\|q\| = \sqrt{q_0^2 + \boldsymbol{q}^T\boldsymbol{q}} = 1$ can be used to rotate a vector $\boldsymbol{p}$ around the normalized axis $\boldsymbol{\alpha}$ with the angle φ by applying the product

$$p' = qp\bar{q}, \qquad q = \begin{pmatrix} \cos(\varphi/2) \\ \sin(\varphi/2)\boldsymbol{\alpha} \end{pmatrix}. \tag{5}$$

The non-bold notation of p denotes that the vector is embedded into a quaternion with vector part $\boldsymbol{p}$ and scalar part 0. This product can also be written as the quaternion matrix-vector product

$$p' = \hat{\mathcal{Q}}(q)^T\mathcal{Q}(q)p = \mathcal{Q}(q)\hat{\mathcal{Q}}(q)^T p = \left(\begin{array}{c|c} 1 & \boldsymbol{0}^T \\ \hline \boldsymbol{0} & \boldsymbol{R} \end{array}\right) p. \tag{6}$$

Carrying out the matrix product results in the well known Rodriguez formula which relates the rotation matrix $\boldsymbol{R}$ and the quaternion

$$\boldsymbol{R}(q) = 2\boldsymbol{q}\boldsymbol{q}^T + (q_0^2 - \boldsymbol{q}^T\boldsymbol{q})\boldsymbol{1} + 2q_0[\boldsymbol{q}]^\times. \tag{7}$$

Derivatives: For our rod model we will define constraints as functions of quaternions. In order to enforce them in a position-based fashion we will need some derivatives of quaternion functions which will be computed here. First we need the derivatives of the quaternion product pq which can be done easily using the quaternion matrix

$$\frac{\partial}{\partial q}(pq) = \frac{\partial}{\partial q}(\mathcal{Q}(p)q) = \mathcal{Q}(p), \tag{8}$$

$$\frac{\partial}{\partial p}(pq) = \frac{\partial}{\partial p}(\hat{\mathcal{Q}}(q)p) = \hat{\mathcal{Q}}(q). \tag{9}$$

Further we will need the derivative of a rotated vector $qp\bar{q}$ w.r.t. q. Therefore we take the derivative of (7).

$$\begin{aligned} \frac{\partial}{\partial q}\boldsymbol{R}(q)\boldsymbol{p} &= \frac{\partial}{\partial q}\left(2\boldsymbol{q}\boldsymbol{q}^T\boldsymbol{p} + q_0^2\boldsymbol{p} - \boldsymbol{p}\boldsymbol{q}^T\boldsymbol{q} + 2q_0\boldsymbol{q}\times\boldsymbol{p}\right) \\ &= \left(\begin{array}{c|c} 2q_0\boldsymbol{p} - 2\boldsymbol{p}\times\boldsymbol{q} & 2\boldsymbol{q}^T\boldsymbol{p}\boldsymbol{1} + 2\boldsymbol{q}\boldsymbol{p}^T - 2\boldsymbol{p}\boldsymbol{q}^T - 2q_0[\boldsymbol{p}]^\times \end{array}\right) \end{aligned} \tag{10}$$

At first glance this matrix may seem quite arbitrary, but it is actually the lower 3×4 block of the quaternion matrix $2\hat{\mathcal{Q}}(p\bar{q})$ which can be seen by rearranging the quaternion matrix.

$$\begin{aligned} \hat{\mathcal{Q}}(p\bar{q}) &= \left(\begin{array}{c|c} \cdot & \cdot \\ \hline q_0\boldsymbol{p} - \boldsymbol{p}\times\boldsymbol{q} & \boldsymbol{p}^T\boldsymbol{q}\boldsymbol{1} - [q_0\boldsymbol{p} - \boldsymbol{p}\times\boldsymbol{q}]^\times \end{array}\right) \\ &= \left(\begin{array}{c|c} \cdot & \cdot \\ \hline q_0\boldsymbol{p} - \boldsymbol{p}\times\boldsymbol{q} & \boldsymbol{p}^T\boldsymbol{q}\boldsymbol{1} - q_0[\boldsymbol{p}]^\times + \boldsymbol{q}\boldsymbol{p}^T - \boldsymbol{p}\boldsymbol{q}^T \end{array}\right) \end{aligned}$$

Here we used the formula $[\boldsymbol{p}\times\boldsymbol{q}]^\times = \boldsymbol{q}\boldsymbol{p}^T - \boldsymbol{p}\boldsymbol{q}^T$ which is a matrix version of the Grassmann identity. Now a comparison with (10) shows that

$$2\hat{\mathcal{Q}}(p\bar{q}) = \left(\begin{array}{c} \cdot \\ \hline \frac{\partial}{\partial q}\boldsymbol{R}(q)\boldsymbol{p} \end{array}\right). \tag{11}$$

We will also need the temporal derivative of a quaternion that describes a frame which rotates with angular velocity $\boldsymbol{\omega}$ relative to a world coordinate frame. This derivative is found to be

$$\dot{q} = \tfrac{1}{2}q\omega \tag{12}$$

which is a standard result from rigid-body dynamics and is derived in detail in [Bar97] or [SM06]. Here $\boldsymbol{\omega}$ is expressed in body frame coordinates. It can be transformed to world coordinates by the rotation $\omega^{(w)} = \bar{q}\omega q$ which leads to

$$\dot{q} = \tfrac{1}{2}\omega^{(w)}q. \tag{13}$$

4. PBD with quaternions and vector constraints

In standard PBD all simulated objects consist of point particles. In order to simulate rods we enhance the algorithm so that objects are also described by orientations which are represented as unit quaternions. All interactions between particles and orientations are modeled as constraints. They are scalar or vector functions which take positions and quaternions as arguments. Our simulations system is characterized by the following attributes.

particle $i \in [1,...,N]$		orientation $j \in [1,...,N_q]$	
position	$\boldsymbol{x}_i$	quaternion	q_j
velocity	$\boldsymbol{v}_i$	angular velocity	$\boldsymbol{\omega}_j$
inverse mass	w_i	inertia matrix	$\boldsymbol{I}_j$

constraint $k \in [1,...,M]$	
cardinalities	n_k, m_k
vector function	$\boldsymbol{C}_k : \mathbb{R}^{3n_k} \times \mathbb{H}^{m_k} \to \mathbb{R}^l$
set of particle indices	$\mathcal{I}_k^p = \{i_1,...,i_{n_k}\}, i_a \in [1,...,N]$
set of orientation indices	$\mathcal{I}_k^q = \{i_1,...,i_{m_k}\}, i_a \in [1,...,N_q]$
stiffness parameter vector	$\boldsymbol{k} = (k_1,...,k_l)$, $k_a \in [0,1]$
type	unilateral or bilateral

The linear motion of the particles is described by the position $\boldsymbol{x}$, velocity $\boldsymbol{v}$ and inverse mass w. Analogously the rotational motion of the orientations is described by a unit quaternion q, the angular velocity $\boldsymbol{\omega}$ and the inertia matrix $\boldsymbol{I}$. The constraints are vector functions which map n positions and m quaternions to a vector in $\mathbb{R}^l$. Therefore each constraint is characterized by a set of particle indices $\mathcal{I}^p$ with cardinality n and a set of orientation indices $\mathcal{I}^q$ with cardinality m. The stiffness of each constraint component can be tuned with a stiffness parameter k_a with values in the range from 0 (constraint is not enforced) to 1 (constraint is enforces as strictly as possible). When each component of $\boldsymbol{C}$ has to satisfy the inequality $\boldsymbol{C}_a(\boldsymbol{x}_{i_1},...,\boldsymbol{x}_{i_{n_j}}) \geq 0$ its type is unilateral and it is bilateral when the equality $\boldsymbol{C}_a(\boldsymbol{x}_{i_1},...,\boldsymbol{x}_{i_{n_j}}) = 0$ has to be satisfied.

Given this information for an initial time t_0 and a time step Δt the simulation works as described in algorithm 1. The main loop can be divided into three steps. The first one computes predictions of positions and quaternions with an explicit Euler scheme. Thereby the constraints are ignored. The second step solves the constraints iteratively and the predicted positions and quaternions are updated so that constraint violation is minimized. In the third step velocities are updated. We will explain these steps in more detail during this section.

Initialization: First the positions, quaternions, linear and angular velocities, inverse masses and inertia matrices are initialized (lines 1-4). Angular motion is described in the frames which are defined by the quaternions so that all inertia matrices are diagonal and constant over time.

Algorithm 1 Position and Orientation Based Dynamics

1: **for all** particles i **do**
2: initialize $\boldsymbol{x}_i = \boldsymbol{x}_i^0$, $\boldsymbol{v}_i = \boldsymbol{v}_i^0$, $w_i = 1/m_i$
3: **for all** orientations j **do**
4: initialize $q_j = q_j^0$, $\boldsymbol{\omega_j} = \boldsymbol{\omega_j^0}$, $\boldsymbol{I}_j = \boldsymbol{I}_j$
5: **loop**
6: **for all** particles i **do** $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta t w_i \boldsymbol{f}_{ext}(\boldsymbol{x}_i)$
7: **for all** particles i **do** $\boldsymbol{p}_i \leftarrow \boldsymbol{x}_i + \Delta t \boldsymbol{v}_i$
8: **for all** orientations j **do**
9: $\boldsymbol{\omega_j} \leftarrow \boldsymbol{\omega_j} + \Delta t \boldsymbol{I}_j^{-1}(\boldsymbol{\tau}_j - \boldsymbol{\omega_j} \times (\boldsymbol{I}_j \boldsymbol{\omega_j}))$
10: **for all** orientations j **do**
11: $u_j \leftarrow q_j + 0.5 \cdot \Delta t q_j \omega_j$
12: $u_j \leftarrow u_j / \|u_j\|$
13: **for all** particles i **do** genCollConstraints$(\boldsymbol{x}_i \rightarrow \boldsymbol{p}_i)$
14: **loop** solveriteration **times**
15: projectConstraints$(\boldsymbol{C}_1, ..., \boldsymbol{C}_{M+M_{coll}}, \boldsymbol{p}, u)$
16: **for all** particles i **do**
17: $\boldsymbol{v}_i \leftarrow (\boldsymbol{p}_i - \boldsymbol{x}_i)/\Delta t$
18: $\boldsymbol{x}_i \leftarrow \boldsymbol{p}_i$
19: **for all** orientations j **do**
20: $\boldsymbol{\omega}_j \leftarrow \Im[2\bar{q}u/\Delta t]$
21: $q_j \leftarrow u_j$

Prediction: The simulation loop starts at line 5. It begins with computing predictions of the positions and quaternions using a semi-implicit Euler scheme (lines 6-12). Also external forces $\boldsymbol{f}_{ext}$ such as gravity and torques $\boldsymbol{\tau}$ are applied. The numerical integration of angular velocities and quaternions is performed using the Newton-Euler equations which are derived in detail in rigid body dynamics literature [Bar97] or [SM06]. The predictions are not assigned to $\mathbf{x}$ and q directly because the old positions and quaternions will be needed to update the velocities. The predictions are stored as $\boldsymbol{p}$ and u. In line 13 collision detection is performed and additional constraints for resolving collisions are generated. These collision constraints are generated at the beginning of each time step and are used only for one time step.

Correction: The core component of the algorithm is the iterative constraint solver (lines 14-16). It iterates multiple times over all constraints and updates positions and quaternions such that the constraint violation is reduced. Each constraint is considered in isolation. In order to compute correction displacements for the particles and quaternions which are involved in the constraint the function $\boldsymbol{C}$ gets linearized

$$\boldsymbol{C}(\boldsymbol{p} + \Delta\boldsymbol{p}) = \boldsymbol{C}(\boldsymbol{p}) + \nabla_p \boldsymbol{C} \Delta\boldsymbol{p} = 0. \tag{14}$$

Here the Jacobian of $\boldsymbol{C}$ is denoted as $\nabla_p \boldsymbol{C}$. The vector $\boldsymbol{p} = (\boldsymbol{p}_1^T, ..., \boldsymbol{p}_N^T, q_1^T, ..., q_{N_q}^T)^T$ which collects all positions and quaternions is used to simplify notation. This can be done because the quaternions are treated exactly in the same way as the positions. We cannot just use (14) and solve for $\Delta\boldsymbol{p}$ because the dimension of $\Delta\boldsymbol{p}$ will be usually much higher than the dimension of $\boldsymbol{C}$ which means that the problem is under-determined. To solve this problem the displacements are restricted to the derivative direction

$$\Delta\boldsymbol{p} = \boldsymbol{W}(\nabla_p \boldsymbol{C})^T \boldsymbol{\lambda} \tag{15}$$

of the constraint function. They are weighted with the mass/inertia matrix $\text{diag}(w_1\mathbf{1}, ..., w_N\mathbf{1}, \boldsymbol{W}_1, ..., \boldsymbol{W}_{N_q})$ to achieve conservation of linear and angular momentum. Plugging this into (14) and solving for $\boldsymbol{\lambda}$ yields

$$\boldsymbol{\lambda} = -\left(\sum_k (\nabla_{p_k} \boldsymbol{C}) \boldsymbol{W}_k (\nabla_{p_k} \boldsymbol{C})^T \right)^{-1} \boldsymbol{C}(\boldsymbol{p}) \tag{16}$$

where we sum over all positions and quaternions which are involved in the constraint $\boldsymbol{C}$. Plugging this result back into (15) yields the displacement for one particle or quaternion with index i

$$\Delta\boldsymbol{p}_i = -\boldsymbol{W}_i (\nabla_p \boldsymbol{C}_i)^T \left(\sum_k (\nabla_{p_k} \boldsymbol{C}) \boldsymbol{W}_k (\nabla_{p_k} \boldsymbol{C})^T \right)^{-1} \boldsymbol{C}(\boldsymbol{p}). \tag{17}$$

This equation was first presented in [USS14] and is a generalization of the original PBD formula [MHHR07]. It allows the enforcement of vector constraints or multiple scalar constraints at once which leads to faster convergence. If $\boldsymbol{C}$ is a scalar function, (17) becomes the original PBD formula. The quaternions have to be normalized after every update because they only describe proper rotations if they have unit length. Concrete examples how to compute the displacements for rod constraints are presented in section 6.

Velocity update: In the end of the algorithm the corrected positions $\boldsymbol{p}$ and quaternions u are used to update the velocities, positions and quaternions. Therefore we use the discrete time derivative

$$\boldsymbol{v} = \dot{\boldsymbol{x}} \approx (\boldsymbol{x}^{t+\Delta t} - \boldsymbol{x}^t)/\Delta t = (\boldsymbol{p} - \boldsymbol{x})/\Delta t \tag{18}$$

to update linear velocity. In case of angular velocity we use (12), discretize it and solve for $\boldsymbol{\omega}$ yielding

$$\boldsymbol{\omega} = \Im[2\bar{q}\dot{q}] \approx \Im\left[2\bar{q}^t (q^{t+\Delta t} - q^t)/\Delta t\right] = \Im[2\bar{q}u/\Delta t]. \tag{19}$$

5. Cosserat theory

Cosserat rods are described by a smooth curve $\boldsymbol{r}(s) : [s_0, s_1] \rightarrow \mathbb{R}^3$ in 3d space. An orthonormal frame with the basis vectors $\{\boldsymbol{d}_1(s), \boldsymbol{d}_2(s), \boldsymbol{d}_3(s)\}$ is attached to every point of the curve such that $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ span the plane of the rod's cross section and $\boldsymbol{d}_3 = \boldsymbol{d}_1 \times \boldsymbol{d}_2$ is the normal of the cross section (see figure 2). The vectors

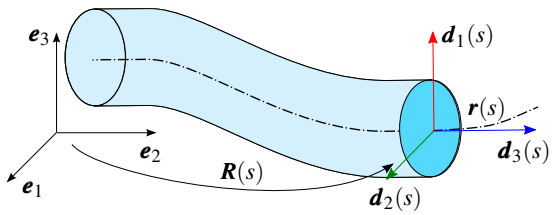

Figure 2: *The geometry of the continuous Cosserat rod model is described by the centerline* $\boldsymbol{r}$ *and the directors* $\boldsymbol{d}_k$.

$\boldsymbol{d}_k$ are also called directors. Given a space fixed world coordinate

system with the basis $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ the directors can be represented as rotated basis vectors $d_k = qe_k\bar{q}$ with the rotation quaternion $q(s)$.

Strain measures for shear and stretch: The tangent vector of the centerline $\partial_s \boldsymbol{r}(s)$ is not necessarily parallel to the normal of the rod's cross section $\boldsymbol{d}_3$. If they are not parallel the rod is subject to shear deformations. If the rod is in the rest configuration the tangent of the centerline has unit length (arc-length or unit speed parametrization). Its length is shortened when the rod gets compressed and it increases when the rod gets stretched. This can be used to define a strain measure $\boldsymbol{\Gamma}$ for the shear and stretch deformations

$$\boldsymbol{\Gamma}(s) = \partial_s \boldsymbol{r}(s) - \boldsymbol{d}_3(s). \tag{20}$$

It measures how much the rod deviates from its non stretched and non sheared rest pose. $\boldsymbol{\Gamma}$ vanishes if the centerline's tangent and the normal of the cross section are parallel, which means no shear is applied, and if the length of the tangent vector is 1 because $\|\boldsymbol{d}_3\| = 1$, which means that the rod is not stretched or compressed. The strain measure $\boldsymbol{\Gamma}$ can also be expressed in the material frame which moves along with the rod and is spanned by the directors in every curve point. Therefore it has to be rotated with the quaternion $\bar{q}(s)$

$$\tilde{\boldsymbol{\Gamma}} = \boldsymbol{R}(q)^T \partial_s \boldsymbol{r} - \boldsymbol{e}_3. \tag{21}$$

Strain measures for bend and twist: Further deformations of the rod are bending and twisting. To define a strain measure for those we use the Darboux vector $\boldsymbol{\Omega}$. It is defined as

$$\boldsymbol{\Omega} = \tfrac{1}{2} \sum_{k=1}^{3} \boldsymbol{d}_k \times \boldsymbol{d}_k', \tag{22}$$

where $'$ denotes the derivative w.r.t. s. This definition is equivalent to

$$\mathbf{d}_k' = \boldsymbol{\Omega} \times \mathbf{d}_k \tag{23}$$

which is proved in the appendix of [USS14]. Equation (23) shows that the Darboux vector is very similar to angular velocity. When a rigid body and a body-fixed frame rotate with angular velocity $\boldsymbol{\omega}$ the temporal derivative of the basis vectors is found to be $\dot{\mathbf{d}}_k = \boldsymbol{\omega} \times \mathbf{d}_k$ (see [SM06]). There $\boldsymbol{\omega}$ is the rate of change of the basis vectors when time is varied. In the same way the Darboux vector $\boldsymbol{\Omega}$ describes the rate of chance of the directors when the curve parameter s is varied. Analogously to angular velocity in (12) the Darboux vector can be expressed in terms of the rotation quaternion

$$\Omega = 2\bar{q}q'. \tag{24}$$

The proof of (24) is the same as for $\boldsymbol{\omega}$ which can be found in [Bar97], except that the temporal derivative ∂_t has to be exchanged with the spatial derivative ∂_s. Notice, that the Darboux vector in (24) is represented in the material frame. This means that the first and second component Ω_1/Ω_2 measure the curvature or bending of the rod in direction of $\boldsymbol{d}_1/\boldsymbol{d}_2$ and Ω_3 measures the rod's twist in direction $\boldsymbol{d}_3$. Now we can define a strain measure for bending and twisting as

$$\Delta\Omega = \Omega - \Omega^0 = 2\left(\bar{q}q' - \bar{q}^0 q'^0\right) \tag{25}$$

where the superscript 0 denotes that the values are taken in the rod's rest shape.

Invariance under rigid body motion: An important feature of the material frame strain measures is that they are invariant under rigid body transformations. When the entire rod gets translated or rotated the strain measures report the same deformation. The translation invariance is obvious because $\boldsymbol{\omega}$ depends only on quaternions and $\tilde{\boldsymbol{\Gamma}}$ takes the derivative of the centerline which eliminates a constant translation vector. When the entire rod gets rotated with the unit quaternion u, the centerline becomes $ur\bar{u}$ and the quaternions become uq. This results in the rotated strain measures

$$\tilde{\Gamma}^* = \overline{(uq)}\partial_s(ur\bar{u})uq - e_3 = \bar{q}\bar{u}u\partial_s r\bar{u}uq - e_3 = \tilde{\Gamma} \tag{26}$$

(we used the relation $\overline{(uq)} = \bar{q}\bar{u}$) and

$$\Omega^* = 2\overline{(uq)}\partial_s(uq) = 2\bar{q}\bar{u}u\partial_s q = \Omega \tag{27}$$

which proves their rotation invariance.

Discrete rods: In force based simulation deformation energies are defined as $E_s = \boldsymbol{\Gamma}^T \boldsymbol{\kappa} \boldsymbol{\Gamma}$ and $E_b = \Delta\boldsymbol{\Omega}^T \boldsymbol{K} \Delta\boldsymbol{\Omega}$ with the stiffness matrices $\boldsymbol{\kappa}$ and $\boldsymbol{K}$. These are used to derive forces and the rod can be discretized with finite elements or finite differences. In contrast to this approach we will use the discretized strain measures $\boldsymbol{\Gamma}$ and $\Delta\boldsymbol{\Omega}$ as vector constraints in PBD. It is also possible to minimize the deformation energies with the position-based solver. But it was pointed out in [USS14] that this approach leads to slower convergence because the required linearization of the purely quadratic energy function is not a good approximation.
To discretize the strain measures we use the staggered grid approach which was successfully used by [ST07, BWR*08, LLA11, USS14]. The rod is sampled as piecewise linear elements and the mass is lumped in particles at the endpoints of each element. The origin of a director frame is attached to the midpoint of each line element as depicted in figure 3. The particles are numbered with

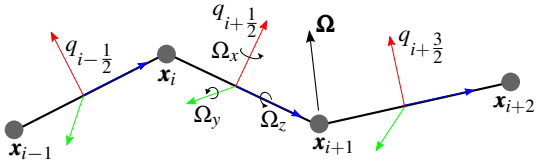

Figure 3: *The geometry of the discrete rod.*

integer indices and the frames or quaternions with half-integer indices. Such a line segment with 2 particles and a quaternion will be called a rod element. Two adjacent rod elements share one particle meaning that a rod which is formed by N elements is described by $N+1$ particles and N quaternions. The discrete strain measures are derived using finite differences as proposed in [LLA11]. The discrete tangent vector is $\partial_s \boldsymbol{r}(s) \approx \frac{1}{l}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i)$ where l denotes the length of the rod element. In order to measure bend and twist at the particle positions we need values of the quaternions there. They can be obtained by interpolation of the two adjacent quaternions. This interpolation is not unique and several choices are presented in [LLA11]. The simplest way is to use the arithmetic mean of the adjacent quaternions. This results in the discrete Darboux vector

$$\boldsymbol{\Omega} \approx \tfrac{1}{l}\Im\left((\bar{q}_{i+\frac{1}{2}} + \bar{q}_{i-\frac{1}{2}})(q_{i+\frac{1}{2}} - q_{i-\frac{1}{2}})\right) = \tfrac{2}{l}\Im\left(\bar{q}_{i-\frac{1}{2}} q_{i+\frac{1}{2}}\right)$$

because $\Im(q\bar{q}) = \mathbf{0}$ for a unit quaternion q. Taking only the imaginary part is important in the discrete scenario because the scalar part does not necessarily vanish as in the continuous case. For simplicity we assume that all rod elements have equal length l. The discrete Darboux vector has a simple geometric interpretation. It is the imaginary part of the rotation quaternion that realizes a rotation from the frame $i-\frac{1}{2}$ to the next frame $i+\frac{1}{2}$. Finally we obtain the discrete strain measures

$$\boldsymbol{\Gamma}_{i+\frac{1}{2}} \approx \tfrac{1}{l}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) - \Im\left(q_{i+\frac{1}{2}} e_3 \bar{q}_{i+\frac{1}{2}}\right), \tag{28}$$

$$\Delta\boldsymbol{\Omega}_i \approx \tfrac{2}{l}\Im\left(\bar{q}_{i-\frac{1}{2}} q_{i+\frac{1}{2}} - \bar{q}^0_{i-\frac{1}{2}} q^0_{i+\frac{1}{2}}\right). \tag{29}$$

Alternative interpolation: We also tried to use a different interpolation of the quaternions. The Darboux vector

$$\boldsymbol{\Psi} = \tfrac{2}{l}\Im\left(\bar{q}_{i-\frac{1}{2}} q_{i+\frac{1}{2}}\right) / \Re\left(\bar{q}_{i-\frac{1}{2}} q_{i+\frac{1}{2}}\right) \tag{30}$$

is equivalent to the modified Darboux vector used in [USS14]. It points in the same direction as the Darboux vector $\boldsymbol{\Omega}$. But it has the useful property that its magnitude scales with $\tan(\theta/2)$ where θ is the angle between the adjacent frames. It follows directly from the facts that $\sin(\theta/2) = |\Im(\bar{q}_{i-\frac{1}{2}} q_{i+\frac{1}{2}})|$ and $\cos(\theta/2) = \Re(\bar{q}_{i-\frac{1}{2}} q_{i+\frac{1}{2}})$. This means that $\boldsymbol{\Psi}$ and therefore deformation energy diverges when $\theta = \pi$. This helps to avoid the undesired and unphysical effect of completely kinked rods. Further $\boldsymbol{\Psi}$ is a monotonically increasing function which means that the relative rotation with $\theta = 0$ becomes unique so that there is one unique rest pose. From a theoretical point of view the modified Darboux vector is the better choice. But our experiments showed that the scaling has no influence on the correction displacements and the rest pose of the non-modified Darboux vector can be made unique with a simple quaternion trick (see section 6). Therefore we will only discuss constraints which are based on the non-modified Darboux vector $\boldsymbol{\Omega}$ in the remainder of this paper. The projection formulas of the modified Darboux vector constraint can be found in appendix A.

6. Rod constraints

To simulate rods in PBD we use the discrete strain measures as vector constraints. The stretch and shear constraint $\boldsymbol{C}_s$ minimizes the stretch and shear for a rod element which is formed by two adjacent particles with positions $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and the quaternion q between them. The bend and twist constraint $\boldsymbol{C}_b$ couples two adjacent rod elements with quaternions q and u and minimizes bend and twist between them.

$$\boldsymbol{C}_s(\boldsymbol{p}_1, \boldsymbol{p}_2, q) = \tfrac{1}{l}(\boldsymbol{p}_2 - \boldsymbol{p}_1) - \boldsymbol{R}(q)\boldsymbol{e}_3 = 0 \tag{31}$$

$$\boldsymbol{C}_b(q, u) = \Im\left(\bar{q}u - \bar{q}^0 u^0\right) = \boldsymbol{\Omega} - s\boldsymbol{\Omega}^0 = 0 \tag{32}$$

$$s = \begin{cases} +1 & \text{for } |\boldsymbol{\Omega} - \boldsymbol{\Omega}^0|^2 < |\boldsymbol{\Omega} + \boldsymbol{\Omega}^0|^2 \\ -1 & \text{for } |\boldsymbol{\Omega} - \boldsymbol{\Omega}^0|^2 > |\boldsymbol{\Omega} + \boldsymbol{\Omega}^0|^2 \end{cases} \tag{33}$$

Due to the fact that two quaternions $+q$ and $-q$ represent the same rotation, the rest pose of the bend and twist constraint is not unique. Therefore $+\boldsymbol{\Omega}^0$ and $-\boldsymbol{\Omega}^0$ are both valid rest poses. To avoid artifacts it is important that the quaternions are displaced towards the nearest rest pose. This can be achieved by setting the sign factor $s = +1$ if $\boldsymbol{\Omega}$ is nearer to $\boldsymbol{\Omega}^0$ and $s = -1$ if $\boldsymbol{\Omega}$ is nearer to $-\boldsymbol{\Omega}^0$.

Stretch and shear displacements: To obtain the correction displacements, the derivatives with respect to the involved positions and quaternions are needed. For the stretch shear constraint they are found to be

$$\nabla_{\boldsymbol{p}_1}\boldsymbol{C}_s = -\nabla_{\boldsymbol{p}_2}\boldsymbol{C}_s = -\tfrac{1}{l}\mathbf{1}_{3\times 3}, \tag{34}$$

$$\nabla_q \boldsymbol{C}_s = 2\left(\ \Im(e_3\bar{q})\ \middle|\ \Re(e_3\bar{q})\mathbf{1}_{3\times 3} - [\Im(e_3\bar{q})]^\times\ \right). \tag{35}$$

The derivative w.r.t. q is the lower 3×4 block of $2\hat{\mathcal{Q}}(e_3\bar{q})$ which follows directly from (11). To get the corrections of the positions and quaternions we need to plug these results in (17). Therefore we need the products $(\nabla\boldsymbol{C})\boldsymbol{W}(\nabla\boldsymbol{C})^T$. To simplify things we use scalar weights w_q instead of inertia matrices $\boldsymbol{W}_q$ which has not much impact on the visual results. Then we only have to compute the products $(\nabla\boldsymbol{C})(\nabla\boldsymbol{C})^T$. They can be found with a simple trick: we can compute the product $4\hat{\mathcal{Q}}(e_3\bar{q})\hat{\mathcal{Q}}(e_3\bar{q})^T$ instead which contains our desired result in the lower right 3×3 block. Due to the fact that the unit quaternions form a group, the quaternion $z = e_3\bar{q}$ has unit length. Rearranging the quaternion matrix product leads to

$$\hat{\mathcal{Q}}(z)\hat{\mathcal{Q}}(z)^T = \hat{\mathcal{Q}}(z)\hat{\mathcal{Q}}(\bar{z}) = \hat{\mathcal{Q}}(z\bar{z}) = \hat{\mathcal{Q}}(1) = \mathbf{1}_{4\times 4}. \tag{36}$$

This means that we do not have to compute a matrix inversion for every constraint in each time step. We only have to compute the inverse of the matrix $(w_1 + w_2 + w_q)\mathbf{1}$. The notation of the correction displacements can be simplified by noticing that $(\nabla\boldsymbol{C}_s)^T\boldsymbol{C}_s = \hat{\mathcal{Q}}(e_3\bar{q})^T C_s = C_s q\bar{e}_3$ where we embedded the constraint vector into a quaternion with scalar part 0. The vanishing scalar part cancels out the first column of the quaternion matrix and the displacements can be written as quaternion products. This results in the displacements

$$\boxed{\begin{aligned} \Delta\boldsymbol{p}_1 &= +\tfrac{w_1 l}{w_1+w_2+4w_q l^2}\left(\tfrac{1}{l}(\boldsymbol{p}_2 - \boldsymbol{p}_1) - \boldsymbol{d}_3\right), \\ \Delta\boldsymbol{p}_2 &= -\tfrac{w_2 l}{w_1+w_2+4w_q l^2}\left(\tfrac{1}{l}(\boldsymbol{p}_2 - \boldsymbol{p}_1) - \boldsymbol{d}_3\right), \\ \Delta q &= +\tfrac{w_q l^2}{w_1+w_2+4w_q l^2}\left(\tfrac{1}{l}(\boldsymbol{p}_2 - \boldsymbol{p}_1) - \boldsymbol{d}_3\right) q\bar{e}_3. \end{aligned}} \tag{37}$$

Bend and twist displacements: For the bend and twist constraint we can calculate the derivatives of $\bar{q}u$ and then take only the lower 3×4 block of the results. This is equivalent to taking the derivative of the imaginary part. Hence we get $\nabla_u(\bar{q}u) = \mathcal{Q}^T(q)$ and $\nabla_q(\bar{q}u) = \nabla_q(\hat{\mathcal{Q}}(u)\bar{q}) = \hat{\mathcal{Q}}(u)\cdot\ \mathrm{diag}(1,-1,-1,-1)$ leading to the derivatives

$$\nabla_u \boldsymbol{C}_b = +\left(\ -\boldsymbol{q}\ \middle|\ q_0\mathbf{1}_{3\times 3} - [\boldsymbol{q}]^\times\ \right), \tag{38}$$

$$\nabla_q \boldsymbol{C}_b = -\left(\ -\boldsymbol{u}\ \middle|\ u_0\mathbf{1}_{3\times 3} - [\boldsymbol{u}]^\times\ \right). \tag{39}$$

The products $(\nabla\boldsymbol{C})(\nabla\boldsymbol{C})^T$ are found to be $\mathbf{1}_{3\times 3}$ with the same trick as above. Therefore we only have to notice that both derivatives are lower 3×4 blocks of the quaternion matrix $\mathcal{Q}^T$. Again we do not have to compute any matrix inversions. Here it is also possible to simplify the notation of the displacements by embedding $\boldsymbol{C}_b$ into a quaternion with scalar part 0 and multiplying it with the full quaternion matrix. This leads to the correction displacements

$$\boxed{\begin{aligned} \Delta q &= +\tfrac{w_q}{w_q+w_u} u(\Omega - s\Omega^0), \\ \Delta u &= -\tfrac{w_u}{w_q+w_u} q(\Omega - s\Omega^0). \end{aligned}} \tag{40}$$

These displacements are easy to implement and cheap to compute. The main effort is computing the quaternion products and normalizations. Due to the fact that normalizations can be computed very efficiently on modern graphics hardware (see e.g. [nVi]) or CPUs using the SSE reciprocal square root instruction [Int] our technique achieves high performance. To speed up simulations of inextensible rods our constraints can be easily combined with LRA (long-range-attachment) constraints [KCMF12].

7. Implementation details

We implemented our method in C++ and the presented scenes are rendered with OpenGL. All experiments and benchmarks were performed on a single core of an Intel Core i5 3450 CPU which runs at 3.1 GHz. To efficiently detect collisions we implemented the spatial hashing algorithm by Teschner et al. [THM*03]. Collisions are resolved with edge-edge-distance constraints [Ben]. Further we use the friction model and the approximate shock propagation method [MMCK14] to stably simulate large numbers of colliding rod elements like in figure 1.

The quaternions are initialized such that the directors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ lie on the principal axes of the cross section of the rod. This allows us to simulate anisotropic bending stiffness. For rods with isotropic cross section the orientation of $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ is arbitrary. But in order to avoid flipping artifacts the initial quaternions should be chosen such that they minimize the Darboux vector. This can be achieved using parallel transport [BWR*08]. The inverse mass w of the particles is either chosen to be 0 if the particle is fixed or 1 for moving particles. In the same way we set the scalar weight factors w_q of the quaternions to 0 for fixed frames and 1 for moving ones. In the presented examples we used a time step size of 10ms.

Solving the constraints in a sequential order from one end of the rod to the other one can lead to instabilities as pointed out by [USS14]. To overcome this problem they used a bilateral interleaving order, which we also use. It is depicted in figure 4.

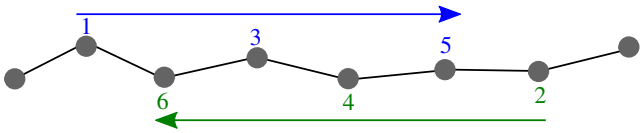

Figure 4: *Bilateral interleaving ordering for fast and stable constraint solving.*

8. Results

We compare our method to two real-time rod and strand simulation methods in terms of visual quality and computational performance.

Comparison with position-based elastic rods: The first one is the position-based elastic rod (PBER) model [USS14] which is most closely related to our method. We implemented their constraint projections exactly as described in [USS14] and the supplemental document. Further we repeated their benchmark experiment

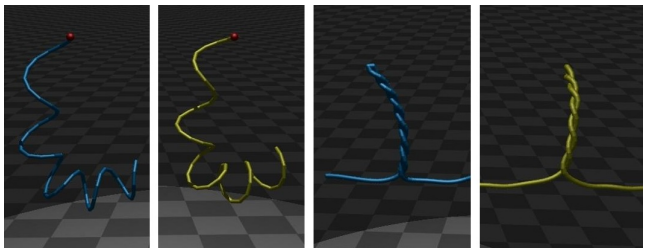

Figure 5: *Our method (blue) achieves the same visual quality as position-based elastic rods (yellow). Torsion effects like curls (left) and coupling of bend and twist are reproduced accurately. The latter is essential for looping phenomena (right) which occur when both ends are clamped and one end is twisted.*

where they simulated a rod with 30 discrete elements and 10 solver iterations. Our PBER implementation requires 1.1ms per frame, which is nearly the same as reported by the authors (1.06ms).
We performed various experiments to show that the visual results of our method are nearly identical to PBER. Some of them are depicted in figure 5 and they are presented in more detail in the accompanying video. Further we compare the computation times per frame of both methods in table 1. Due to the fact that our method requires only 2 constraints per rod element, it converges with much less iterations than PBER which needs 4 constraints per element. Therefore we compare the frame times once with the same number of solver iterations (scene 1). In further experiments (scenes 2 and 3) we tuned the number of iterations such that we achieved nearly the same visual results. With the same number of iterations our method is nearly 12 times faster than PBER. This can be simply explained by comparing the number of arithmetic operations per constraint projection. The PBER bending-twisting constraint couples 5 (ghost) particles and its projection requires roughly one order of magnitude more arithmetic operations than projecting both of our constraints. Additionally our method requires only a fraction of the iterations to produce nearly the same visual results as PBER. This leads to 60-70 times faster computation times per frame.

Table 1: *Computation times per frame for rods with 50 elements. The number in parentheses denotes the number of solver iterations. The scenes are shown in more detail the accompanying video.*

	Scene 1	Scene 2	Scene 3
our method	0.14ms (10)	0.03ms (2)	0.14ms (10)
PBER [USS14]	1.65ms (10)	2.16ms (18)	9.18ms (80)
PBD	0.16ms (10)	0.26ms (18)	1.32ms (80)

Another advantage of our method is that we do not have to worry about unphysical behavior of the system which can be introduced by ghost points. Because of the translation and rotation invariance of the constraints the position-based solver guarantees conservation of linear and angular momentum (see [MHHR07]).

Comparison with position-based dynamics: Secondly we compare our method to standard PBD with distance and line-line bending constraints [MHHR07]. This method is not able to produce

twisting effects, but it is one of the fastest methods for simulating bending resistant strands and is used in many real-time hair simulation frameworks. This comparison is used to underline the high performance of our approach. We disable torsion stiffness by setting the third component of the bending and torsion stiffness factor to zero. Then our system behaves like the standard PBD strand but is again a lot stiffer. This effect is shown in detail in the accompanying video. In terms of computational costs our method is as fast as standard PBD without torsion effects when the same number of iterations is used (see table 1). In contrast to normal line-line bending constraints quaternion constraints do not require any inverse trigonometric functions, which are much more expensive than the reciprocal square root intrinsics.

Robustness: To show the robustness of our method we start the simulation from a random initial pose. This experiment was inspired by [USS14]. As shown in figure 6 the rod stably converges to the rest pose in about 100 frames.

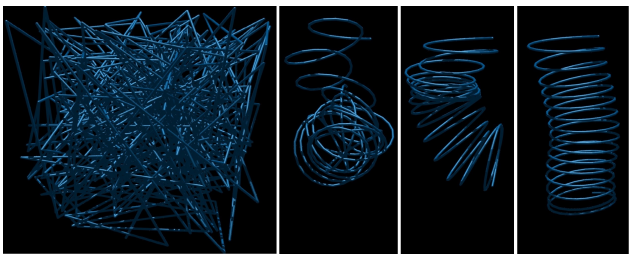

Figure 6: *Started from a random initial pose the spiral stably recovers its rest shape within seconds.*

Figure 1 shows that our method can robustly handle complex scenes. The Slinky spiral consists of 50 curls and is discretized with 1000 elements. Further, friction, anisotropic bending stiffness and robust collision handling are essential in this scene. It was run with 50 solver iterations and took 7*ms* (without collision detection) per frame on a single core of a Core i5 CPU.

9. Conclusion and future work

We presented a fast and robust position-based method for elastic rod simulation by introducing orientation constraints based on quaternions. This allows us to directly minimize the discrete strain measures from Cosserat theory without the need for auxiliary constructs like ghost points. We showed that the displacement formulas are simple algebraic expressions which can be computed very efficiently. An important result is that quaternion constraints can be solved with standard PBD methods. Only few lines of additional code are required in the prediction and velocity update steps. Consequently our method can be implemented with the same ease as standard PBD and can be easily integrated into existing position-based dynamics solvers.

As a position-based method it is unconditionally stable and large time steps can be used. Further it can be highly parallelized and easily coupled with other systems using constraints. But it also inherits all limitations of PBD. The main limitation is that PBD can only produce visually plausible animations but no physically accurate simulations. Another limitation is that material stiffness depends on the time step size, the number of solver iterations and the sampling density of the rod. In real-time physics and animation where computational performance is more important than physical accuracy the drawbacks of position-based methods can usually be tolerated. Material stiffness can be tuned because the parameters it depends on are typically constant.

However the limitations give useful hints for possible future work. We expect that most of the drawbacks can be overcome by using the projective dynamics solver [BML*14], which actually solves an implicit Euler integration scheme. Together with the deformation energies E_s and E_b from section 5 it should result in physically accurate simulations, in which material stiffness is independent of the time step size and the rod sampling. The required projection operators are very similar to our displacement formulas. Being independent of the rod sampling would also allow for adaptive refinement as described by Spillmann and Teschner [ST08].

Another possibility for future research is the combination of our constraints with the tridiagonal matrix solver of Han and Harada [HH13]. Their solver can be interpreted as PBD, but they solve all distance constraints of one strand as a single vector constraint. A key observation is that the matrix in eq. (16) is tridiagonal for a serial chain of particles connected by distance constraints. A simple direct solver can be used to solve for $\boldsymbol{\lambda}$ and all constraints can be satisfied in one iteration. Our rod constraints are more complex than distance constraints, but due to the chain topology the matrix in eq. (16) will be band diagonal. This idea could lead to further speedup and guarantees inextensibility of the rod.

Finally we are optimistic that other areas besides rod simulation can benefit from the possibility to easily solve quaternion constraints in PBD. For instance they could be used to couple rigid bodies. Further we have done first experiments on using quaternion constraints to simulate thin shells and 3d deformable bodies with promising results.

References

[BAC*06] BERTAILS F., AUDOLY B., CANI M.-P., QUERLEUX B., LEROY F., LÉVÊQUE J.-L.: Super-helices for predicting the dynamics of natural hair. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 1180–1187. 2

[Bar97] BARAFF D.: An introduction to physically based modeling: rigid body simulation part 1: unconstrained rigid body dynamics. *SIGGRAPH Course Notes* (1997). 4, 5, 6

[BAV*10] BERGOU M., AUDOLY B., VOUGA E., WARDETZKY M., GRINSPUN E.: Discrete viscous threads. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 116. 3

[Ben] BENDER J.: Position based dynamics library. URL: https://github.com/janbender/PositionBasedDynamics. 8

[Ber09] BERTAILS F.: Linear time super-helices. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 417–426. 2

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-based simulation of continuous materials. *Computers & Graphics 44* (2014), 1–10. 3

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 154. 9

[BMOT13] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M.:

Position-based methods for the simulation of solid objects in computer graphics. *EUROGRAPHICS 2013 State of the Art Reports* (2013). 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 43–54. 3

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete elastic rods. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 63. 3, 6, 8

[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* (2005), ACM, pp. 153–160. 3

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid-body dynamics. *Computer Animation and Virtual Worlds* (2014). 3

[GS07] GRÉGOIRE M., SCHÖMER E.: Interactive simulation of one-dimensional flexible parts. *Computer-Aided Design 39*, 8 (2007), 694–707. 2

[Had06] HADAP S.: Oriented strands: dynamics of stiff multi-body system. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), Eurographics Association, pp. 91–100. 3

[HH13] HAN D., HARADA T.: Tridiagonal matrix formulation for inextensible hair strand simulation. In *Workshop on Virtual Reality Interaction and Physical Simulation* (2013), Bender J., Dequidt J., Duriez C., Zachmann G., (Eds.), The Eurographics Association. doi:10.2312/PE.vriphys.vriphys13.011-016. 3, 9

[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* (2013), ACM, pp. 63–71. 3

[Int] INTEL: Intel intrinsics guide. URL: https://software.intel.com/sites/landingpage/IntrinsicsGuide/. 8

[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 11th ACM SIGGRAPH/Eurographics conference on Computer Animation* (2012), Eurographics Association, pp. 305–310. 3, 8

[LL09] LANG H., LINN J.: *Lagrangian field theory in space-time for geometrically exact Cosserat rods*. ITWM, Fraunhofer Inst. Techno-und Wirtschaftsmathematik, 2009. 3

[LLA11] LANG H., LINN J., ARNOLD M.: Multi-body dynamics simulation of geometrically exact cosserat rods. *Multibody System Dynamics 25*, 3 (2011), 285–312. 1, 2, 3, 6

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. In *ACM transactions on graphics (TOG)* (2011), vol. 30, ACM, p. 92. 3

[MCK12] MÜLLER M., CHENTANEZ N., KIM T.-Y.: Fast simulation of inextensible hair and fur. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2012), Eurographics Association. 3

[MCKM14] MÜLLER M., CHENTANEZ N., KIM T.-Y., MACKLIN M.: Strain based dynamics. In *Symposium on Computer Animation* (2014), pp. 149–157. 3

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 1, 3, 5, 8

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 471–478. 3

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 104. 3

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 153. 3, 8

[MMS15] MICHELS D. L., MUELLER J. P. T., SOBOTTKA G. A.: A physically based approach to the accurate simulation of stiff fibers and stiff fiber meshes. *Computers & Graphics 53B* (Dec. 2015), 136–146. 3

[Mue08] MUELLER M.: Hierarchical Position Based Dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)* (2008), The Eurographics Association. doi:10.2312/PE/vriphys/vriphys08/001-010. 3

[nVi] NVIDIA: Cuda c programming guide. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#arithmetic-instructions. 8

[Pai02] PAI D. K.: Strands: Interactive simulation of thin solids using cosserat models. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 347–352. 2

[RKN10] RUNGJIRATANANON W., KANAMORI Y., NISHITA T.: Chain shape matching for simulating complex hairstyles. In *Computer graphics forum* (2010), vol. 29, Wiley Online Library, pp. 2438–2446. 3

[SBT07] SPILLMANN J., BECKER M., TESCHNER M.: Non-iterative computation of contact forces for deformable objects. *JOURNAL OF WSCG* (2007), 1–3. 3

[SLF08] SELLE A., LENTINE M., FEDKIW R.: A mass spring model for hair simulation. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 64. 1, 3

[SM06] SCHWAB A., MEIJAARD J.: How to draw euler angles and utilize euler parameters. In *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2006), American Society of Mechanical Engineers, pp. 259–265. 4, 5, 6

[ST07] SPILLMANN J., TESCHNER M.: C o r d e: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 63–72. 2, 6

[ST08] SPILLMANN J., TESCHNER M.: An adaptive contact model for the robust simulation of knots. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 497–506. 3, 9

[Sta09] STAM J.: Nucleus: Towards a unified dynamics solver for computer graphics. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics' 09. 11th IEEE International Conference on* (2009), IEEE, pp. 1–11. 3

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANTES D., GROSS M. H.: Optimized spatial hashing for collision detection of deformable objects. In *VMV* (2003), vol. 3, pp. 47–54. 8

[USS14] UMETANI N., SCHMIDT R., STAM J.: Position-based elastic rods. In *ACM SIGGRAPH 2014 Talks* (2014), ACM, p. 47. 1, 2, 3, 5, 6, 7, 8, 9

Appendix A: Bend and twist with modified Darboux vector:

Using the modified Darboux vector $\mathbf{\Psi}$ the bend-twist constraint is

$$\boldsymbol{C}_b(q,u) = \frac{\Im(\bar{q}u)}{\Re(\bar{q}u)} - \frac{\Im(\bar{q}^0 u^0)}{\Re(\bar{q}^0 u^0)} = \mathbf{\Psi} - \mathbf{\Psi}^0. \tag{41}$$

It is related to the angle θ between the adjacent frames with quaternions q and u by $\cos(\theta/2) = \Re(\bar{q}u) = q^T u$ and $\sin(\theta/2) = |\Im(\bar{q}u)|$. With the results from section 6 this lead to the displacements:

$$\Delta q = +\frac{w_q}{w_q+w_u}\left((q^T u)u + u\Im(\bar{q}u)^T\right)\left(\mathbf{1}_{3\times 3} - \Im(\bar{q}u)\Im(\bar{q}u)^T\right)\boldsymbol{C}_b,$$
$$\Delta u = -\frac{w_u}{w_q+w_u}\left((q^T u)q - q\Im(\bar{q}u)^T\right)\left(\mathbf{1}_{3\times 3} - \Im(\bar{q}u)\Im(\bar{q}u)^T\right)\boldsymbol{C}_b.$$