Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2011)
J. Bender, K. Erleben, and E. Galin (Editors)

Realtime Simulation of Stiff Threads Using Large Timesteps

Nathan Hüsken[†1]

[1]University of Heidelberg

Abstract

A method to simulate suture threads for a microsurgical training simulator is presented. An integration method based on implicit integration is used in order to achieve the high stiffness of suture threads while still using a large time step for keeping the real-time capabilities of the simulation. A key feature of the presented method is the separation of the contact handling and the integration itself. This preserves the real time capability of the simulation even in complex situations that involve many self collisions (e. g. tying knots). The stability and realism of the model are demonstrated by simulating a reef knot, which is one way of tying and holding together two simulated threads. The methods have been developed in cooperation with the VRmagic GmbH in Mannheim.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Microsurgery is a method in which surgical interventions are performed with high precision instruments while being viewed through microscopes. Intensive training is required to develop a high degree of hand dexterity and hand-eye coordination while looking through a microscope, skills necessary for microsurgery. A computer simulation offers opportunities for learning the required skills without the necessity of doing surgery on anaesthetized animals or letting an untrained surgeon operate on a human patient. Suturing of tissues and blood vessels is one part of microsurgical intervention. A realistic simulation of the suture material, the thread, is an important requirement. To create an ideal training situation, the simulation has to be accurate.

In this paper an approach to simulate a thread for suturing based on the mass spring model is introduced. The high stiffness of the thread poses a problem when traditional explicit integrators are used. With explicit integration the timestep with which the simulation is updated has to be inverse proportional to the stiffness of the spring, increasing the computational demands of the simulation. On the other hand implicit integration is stable even for large time steps but requires solving a linear system of size n, where n is the number of mass points representing the thread. This is generally not possible in $O(n)$.

In this paper the resolution of interactions (e .g. thread-thread collisions) between non-adjacent mass points is separated from the integration itself. The integration thereby becomes a banded linear system solvable in $O(n)$. Solving for the contact forces at the collision points is then only a problem of size c where c is the number of contact points. For this, the interactions between contact points are derived from the integration formulas of the integration method, allowing to resolve the contacts in a similar way rigid body contact resolution is normally done. This requires solving a linear complementary problem (LCP).

Additionally, an approximation of the implicit integration is given, simplifying the linear system to a tri-band Matrix. The integration is based on a physical model and allows modelling interactions by applying the corresponding forces.

The correctness of the system is demonstrated by simulating a reef knot, tying together two threads of similar diameter.

2. Related work

Physically-based simulation and animation of deformable objects is an important research area in computer graphics. A good overview off the topic is given in [NMK*06].

† nathan.huesken@ziti.uni-heidelberg.de

© The Eurographics Association 2011.

DOI: 10.2312/PE/vriphys/vriphys11/001-009

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

Most approaches for simulating rods, ropes or threads are modeled representing its centerline as a chain of vertices connected by line segments [ST07], [Pai02], [PK11], [BWR*08]. The explicit representation of the centerline has the advantage of simplifying geometric tests such as collision detection. On the other hand [CJY02] focuses on hair interactions, modeling the hair strands as clusters linked by bending springs. [LWL10] introduces a continuum mechanics based thread which includes rotary inertia shear deformation and torsion. The model allows deformation of the cross section, use of arbitrary cross sections and is realtime capable.

A very simple model for thread simulation called "Follow the leader" is introduced in [BLM04] and embedded in a microsurgical simulator [BMcLS01]. All vertices follow one vertex pulled by the user's input. While the absence of a physical model is apparent, this method allows tying complex knots. Adding other physical effects such as gravity or bending resistance is difficult in this model. Other common approaches for integrating and simulating rods, ropes and threads include finite element [GPL08], finite difference [Kla96] and mass spring models [ST07], [CCK05]. The mass spring model has the advantage of being simple and physically motivated. Physical motivated forces, such as the pulling of the springs, are calculated and applied at the mass points. The mass spring model is also the starting point for this work.

Real world threads can be very stiff, making it necessary to set the spring constants in the thread very high. For the simulation to be stable the timestep squared must be inversely proportional to the stiffness, requiring a very small timestep when using the mass spring method with an explicit integration scheme. This increases the number of necessary computations per frame decreasing the realtime capabilities of the simulation. For example [ST07] uses a timestep of $0.1ms$. The problem can be solved by applying an implicit integration scheme, as proposed in [DSB99] and adapted for the context of cloth simulation in [BW98]. [TGAB08] proposes a deformable model for one-dimensional objects based on elasticity and plasticity theories. The model allows very accurate simulation even of stiff objects utilizing a fast implicit integration method from [HMC01]. However, collisions and contacts are addressed with penalty forces which are not accurate enough for a stable simulation of complex contact situations such as knots.

Implicit integration schemes have the advantage of being stable even for stiff spring forces and large timesteps. Unfortunately implicit integration requires solving a linear system of size n, where n is the number of mass points, in every timestep. This is in general not possible in linear time limiting the realtime capabilities of implicit integration.

The implicit integration couples all mass points of the thread within one integration update. This means that contact points distributed over the thread are also coupled. A force applied to one contact distributes over the thread and effects the situation at all other contacts. A similar situation arises in the simulation of stacked rigid bodies; [Cat05] suggests a Projected Gauss Seidel (PGS) solver. This method is extended and improved in [SHNE10] introducing a Fletcher-Reeves type nonlinear nonsmooth conjugate gradient (NNCG). Both methods have the advantage of benefiting from warmstarting where the solver is initialized with an initial guess close to the correct result. Because the situation does not greatly change between integration updates, the result of the last simulation iteration poses a good candidate for warmstarting.

3. Methods

3.1. Notation and representation of the thread

The thread is modeled with $N \in \mathbb{N}$ mass points connected by $N-1$ springs. The i-th spring connects the i-th mass point with the i+1 mass point. In the following, lower indices are used to index the mass points and springs while in the case of vector quantities the spatial dimensions are indexed by an upper index. The mass points are located at the positions

$$\vec{x}_i(t) := \left(x_i^1(t), x_i^2(t), x_i^3(t)\right)^T, \quad i \in \{1 \dots N\}$$

where t denotes time. The mass of the mass points is denoted by m_i. The state of the threads (position of the mass points) are updated at a time interval of Δt. The springs between the mass points have a spring constant of k_i and a rest length of l_i. Their direction is given by the tangents

$$\vec{t}_i(t) := \frac{\vec{x}_{i+1} - \vec{x}_i}{|\vec{x}_{i+1} - \vec{x}_i|}.$$

The i-th spring applies a force of $\vec{S}_i(t)$ to the i-th mass point and a force of $-\vec{S}_i(t)$ to the i+1 mass point where

$$\vec{S}_i(t) := k_i \cdot \left(\vec{x}_{i+1}(t) - \vec{x}_i(t) - l_i \vec{t}_i(t)\right).$$

To reduce the handling of special cases when deriving the formulas, $\vec{S}_{-1}$ and $\vec{S}_N$ are defined to be 0, allowing to treat the mass points at the endings of the thread the same as all other mass points.

3.2. Algorithm overview

Algorithm 1 states the method in pseudo code. The forces

Algorithm 1 Method overview

loop
  Compute forces acting on mass points.
  Detect collisions and contacts
  Find contact forces resolving contacts
  Integrate threads
end loop

© The Eurographics Association 2011.

acting on the mass points are computed by collecting external and internal forces and by applying the integrator introduced in section 3.3. Knowing these forces, contacts are detected and resolved. The couplings between contacts are found as explained in section 3.6.3 and the contact system is solved using the Fletcher-Reeves type nonlinear non-smooth conjugate gradient (NNCG) introduced in [SHNE10]. The contact forces are applied and the threads are integrated with the integrator from section 3.3.

3.3. Integration

The implicit integration introduced in this paper presumes an integration rule for which there are $\vec{a}_i(t) \in \mathbb{R}^3$ and $b_i(t) \in \mathbb{R}$ such that

$$\Delta\vec{x}_i(t) := \vec{x}_i(t+\Delta t) - \vec{x}_i(t) = \vec{a}_i(t) + b_i(t)\cdot\vec{F}_i(t) \quad (1)$$

at every timestep t. $\vec{F}_i(t)$ denotes the total force acting on the i-th mass point at time t. Most integration procedures fulfill this requirement. To give an example, the semi-implicit Euler integrator is given by

$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t) + \frac{\vec{F}(t)}{m_i}\Delta t$$
$$\vec{x}_i(t+\Delta t) = \vec{x}_i(t) + \vec{v}(t+\Delta t)\Delta t$$

which sets $\vec{a}_i(t)$ and $b_i(t)$ to

$$\vec{a}_i(t) = \vec{v}_i(t)\Delta t \qquad b_i(t) = \frac{(\Delta t)^2}{m_i}.$$

For reasons of readability t will be omitted where the meaning is clear, that is, without ambiguity. The forces operating on the i-th mass point can be split between external forces $\vec{E}_i$ and the spring forces of the adjacent springs. In the following, the external forces applied to a mass point will be denoted by the letter E while for the total force the letter F will be used. Consequently for all $i \in \{1 \dots N\}$

$$\vec{F}_i(t) = \vec{E}_i(t) + \vec{S}_i(t) - \vec{S}_{i-1}(t). \quad (2)$$

In the next step the $\vec{F}_i(t)$ are replaced with

$$\vec{F}_i^{\lambda}(t) := \lambda\vec{F}_i(x(t+\Delta t)) - (1-\lambda)\vec{F}_i(x(t)) \quad (3)$$

where $\lambda \in [0,1]$. Note that for $\lambda = 0$ does not lead to any modification of the original integration procedure while for $\lambda = 1$ an implicit integration procedure is given.

$x(t+\Delta t)$ is not known in advance and has to be solved for. $\vec{F}_i^{\lambda}(t)$ is therefore also not known, and will be approximated by a Taylor expansion:

$$\vec{F}_i^{\lambda}(t) \approx \vec{F}_i(t) + \lambda\sum_{j=1}^{N}\frac{\partial F_i(t)}{\partial\vec{x}_j}\Delta\vec{x}_j$$

For this the derivatives of the components of $\vec{F}_i^{\lambda}(t)$ are needed. $\vec{S}_i$ depends only on $\vec{x}_i$ and $\vec{x}_{i+1}$ and its derivatives are given by

$$\frac{\partial\vec{S}_i}{\partial\vec{x}_i} = -\frac{\partial\vec{S}_i}{\partial\vec{x}_{i+1}} = -k_i\left[1 - s_i\cdot\left(\vec{t}_i^T\vec{t}_i\right)\right] \quad (4)$$

with

$$s_i := \frac{l_i}{|\vec{x}_{i+1} - \vec{x}_i|}.$$

Assuming $\vec{E}_i$ also depends only on neighboring mass points, (1) and (3) can be combined to

$$D_i\Delta\vec{x}_i + O_i^+\vec{x}_{i+1} + O_i^-\vec{x}_{i-1} = \vec{a}_i + b_i\vec{F}_i \quad (5)$$

where

$$D_i := 1 - \lambda b_i\left(\frac{\partial\vec{S}_i}{\partial\vec{x}_i} + \frac{\partial\vec{S}_{i-1}(t)}{\partial\vec{x}_{i-1}} + \frac{\partial\vec{E}_i}{\partial\vec{x}_i}\right)$$
$$O_i^+ := \lambda b_i\left(\frac{\partial\vec{S}_i}{\partial\vec{x}_i} - \frac{\partial\vec{E}_i}{\partial\vec{x}_{i+1}}\right)$$
$$O_i^- := \lambda b_i\left(\frac{\partial\vec{S}_{i-1}}{\partial\vec{x}_{i-1}} - \frac{\partial\vec{E}_i}{\partial\vec{x}_{i-1}}\right)$$

This relation can be written in a linear system

$$M\Delta\vec{x} = \vec{R}$$

where $\Delta\vec{x}$ is the vector of all $\Delta\vec{x}_i$ and $\vec{R} \in \mathbb{R}^{3N}$ and $M \in \mathbb{R}^{3N} \times \mathbb{R}^{3N}$ can be derived from (5). M is a banded matrix with 5 bands. This allows the system to be solved in linear time using LU decomposition [Tho79].

Ensuring Stability: For the system to be stable, all eigenvalues of M have to be bigger than one. Consider the tangent $\vec{t}_i$. It can happen, that $\left\langle\vec{t}_i\middle|\frac{\partial\vec{S}_{i-1}}{\partial\vec{x}_{i-1}}\middle|\vec{t}_i\right\rangle = 0$. If the contribution given by the derivative of $\vec{E}_i$ also vanishes, this results in:

$$\langle\vec{t}|M|\vec{t}\rangle = 1 + \lambda b_i k_i(1 - s_i)$$

If this is smaller or equal to 1, there is an eigenvalue of M which is also smaller or equal to 1. Therefore to ensure numerical stability in all cases it has to be ensured that

$$1 + \lambda b_i + k_i(1 - s_i) \geq 1 + \varepsilon$$

where $\varepsilon \in \mathbb{R}$ is a small number greater 0. This is done by redefining

$$s_i := \min\left(\frac{l_i}{|\vec{x}_{i+1} - \vec{x}_i|}, 1 - \frac{\varepsilon}{\lambda b_i k_i}\right). \quad (6)$$

3.4. Simplified approach

If one assumes that the $\vec{t}_i$ do not change much within one timestep, $\frac{\partial\vec{t}_i}{\partial\vec{x}_i}$ can be set to zero. If the derivatives of the $\vec{E}_i$ are also scalar (diagonal with the same entry on all diagonal elements), D_i, O_i^+ and O_i^- also become scalar, reducing the number of bands from 5 to 1. Because the LU decomposition has a complexity quadratic in the number of bands, this significantly reduces the computational requirements of the

© The Eurographics Association 2011.

integration. In addition the calculations for resolving contacts done in section 3.6 require many multiplications and inversions of these matrices. If the matrices are scalar they can be represented as real numbers, reducing the number of floating point operations from 27 to 1 for every matrix multiplication. While this approximation speeds up calculation, it introduces an error illustrated in figure 1. The correct spring forces are always linear to $\vec{x}_{i+1} - \vec{x}_i$. Their magnitude is proportional to the distance between the current position of the i+1 mass point and the position of the i+1 mass point where the spring would be at rest. $\vec{S}'(t+\Delta t)$ does not consider the updated rest position of the spring at time $t+\Delta t$ but takes the old rest position from time t. The consequences of this error will be discussed in section 4 and 5.

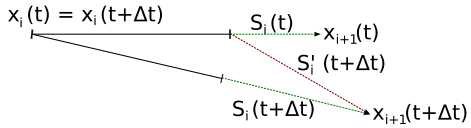

Figure 1: *Error in $\vec{S}_i(t+\Delta t)$ made by the simplified approach. For illustration purposes, it is assumed that $\vec{x}_i(t) = \vec{x}_i(t+\Delta t)$. The correct spring forces at both time t and $t+\Delta t$ are drawn in green while the approximated spring force $\vec{S}'(t+\Delta t)$ is drawn in red. The rest length of the spring along the corresponding tangent is draw in black for both time points.*

3.5. Constraining mass points to positions

When the thread is grasped by forceps or some other medical instrument, it is assumed that the external force applied by the instrument is always big enough to hold the mass point at the position dictated by the user input. Assume the n-th mass point is grabbed. The constraint is enforced by overwriting b_n and $\vec{a}_n$ with

$$b_n(t) = 0 \qquad \vec{a}_n(t) = \vec{x}_{dest} - \vec{x}_n(t)$$

where $\vec{x}_{dest}$ is the position to which the mass point should be constrained. This ensures that the mass point moves to $\vec{x}_{dest}$ independent of any external applied forces.

3.6. Collision handling and response

3.6.1. Collision/Contact detection

The collision detection algorithm detects intersections between pairs of thread segments. The segments are represented as cylinders with a fixed radius. The minimal distance between the centerlines of the cylinders is calculated and tested against the sum of the radii of the two cylinders. If the minimum distance is smaller than the sum of the radii, the closest points on the centerlines are found and represented by the index of the segments in the threads and the positions $\mu_1, \mu_2 \in [0,1]$ on these segments such that the contact points $\vec{c}_i, \ i = \{1,2\}$ are given by

$$\vec{c}_i = (1-\mu_i)\vec{x}_{j_i} + \mu_i \vec{x}_{j_i+1}$$

where $\vec{x}_{j_i}, \vec{x}_{j_i+1}$ are the positions of the adjacent mass points.

The separation vector is defined by

$$\vec{s} := \vec{c}_2 - \vec{c}_1$$

$\vec{s}$ is perpendicular to both colliding segments at the contact point.

A contact is represented as a five-tupel $(i_1, i_2, \mu_1, \mu_2, \vec{s})$ where i_1, i_2 are the indices of the colliding segments and $\mu_1, \mu_2, \vec{s}$ are as defined above.

3.6.2. Contact handling

All contacts are assumed to be inelastic. The contact handling scheme applies forces that maintain non-penetration constraints at all contact points. A contact potentially interacts with two threads. In the following the quantities of the second thread will be denoted with an apostrophe.

The force applied by a contact $c = (i_1, i_2, \mu_1, \mu_2, \vec{s})$ is denoted by $\vec{C}_c$. It results in the forces

$$\Delta\vec{E}_{i_1} = -\vec{C}_c \cdot (1-\mu_1) \qquad \Delta\vec{E}_{i_1+1} = -\vec{C}_c \cdot \mu_1 \tag{7}$$

being applied to the adjacent mass points on the segment of the first thread and the forces

$$\Delta\vec{E}'_{i_2} = \vec{C}_c \cdot (1-\mu_2) \qquad \Delta\vec{E}'_{i_2+1} = \vec{C}_c \cdot \mu_2 \tag{8}$$

being applied to the adjacent mass points on the segment of the second thread. The forces are constant within one timestep, which sets $\frac{\partial \vec{C}_c}{\partial \vec{x}_i} = 0$. Momentum is preserved because these forces sum to zero. In accordance with this, given the movement of the adjacent mass points, the relative contact movement is

$$\begin{aligned} \Delta\vec{x}_c := \ & (1-\mu_2)\cdot\Delta\vec{x}'_{i_2} + \mu_2 \cdot \Delta\vec{x}'_{i_2+1} \\ & -(1-\mu_1)\cdot\Delta\vec{x}_{i_1} - \mu_1 \Delta\vec{x}_{i_1+1}. \end{aligned} \tag{9}$$

The contact force $\vec{C}_c$ is split in the normal force, which is responsible for maintaining the non-penetration condition and the friction force. The normal force is directed in the same direction as the separation distance $\vec{s}$ while the friction force lies in the normal plane of $\vec{s}$. The contact c is resolved when the following conditions are met:

- The relative contact movement $\vec{x}_c$ must resolve the penetration.
- The total relative contact movement must either solve the penetration exactly (no gap between the threads) or (if the threads are pushed apart further) no normal force must be applied at this contact ($\vec{C}_c = 0$).
- The friction force must point in the opposite direction of the relative contact movements and its magnitude must be smaller than the normal force times the coefficient of friction.

© The Eurographics Association 2011.

- If the friction force is big enough to stop the relative movement completely while still fulfilling the last condition (static friction) the resulting relative movement must be zero.

An exact mathematical formulation of these conditions can be found in [SHNE10] and [Cat05].

To solve the problem, the couplings between the contacts (the relation between $\vec{C}_c$ and $\Delta\vec{x}_c$) have to be calculated.

3.6.3. Coupling of contacts

Having a set of contacts $\{c_i\}$, a set of contact forces $\{\vec{C}_{c_i}\}$ must be found fulfilling the conditions from the last section. It is necessary to calculate the change of $\Delta\vec{x}_i$ on one mass point i when a contact force on a different mass point n is applied. Seeing the $\Delta\vec{x}_i$ as a function of $\vec{E}_n$, $\vec{\delta}_{n,i}$ is defined by:

$$\vec{\delta}_{n,i} = \Delta\vec{x}_i(\vec{E}_n) - \Delta\vec{x}_i(0).$$

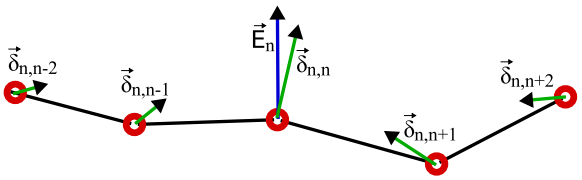

Figure 2: *An external force (blue arrow) results in displacments of all mass points (green arrows).*

Lemma 1: If an external force $\vec{E}_n$ is applied to the n-th mass point, the $\Delta\vec{x}_i$ changes by:

$$\vec{\delta}_{n,n} = \left[D_n + O_n^+ M_{n+1}^- + O_n^- M_{n-1}^+\right]^{-1} b_n \vec{E}_n \tag{10}$$

$$\vec{\delta}_{n,i} = M_i^+ \vec{\delta}_{n,i+1} \quad \text{for} \quad i < n \tag{11}$$

$$\vec{\delta}_{n,i} = M_i^- \vec{\delta}_{n,i-1} \quad \text{for} \quad i > n \tag{12}$$

where

$$M_1^+ := -D_1^{-1} \cdot O_1^+ \qquad M_i^{+1} := -\left[D_i + O_i^- M_{i-1}^+\right]^{-1} O_i^+$$

$$M_N^- := -D_N^{-1} O_N^- \qquad M_i^- := -\left[D_i + O_i^+ M_{i+1}^-\right]^{-1} O_i^-$$

The proof of the lemma is given in the appendix. The coupling $G_{j,i}$ between mass-points i and j on the same thread is defined by

$$\vec{\delta}_{j,i} = G_{j,i} \cdot \vec{E}_j$$

which is a linear equation because equations (10) - (12) are linear. Assume two contacts $c = (i_1, i_2, \mu_1, \mu_2, \vec{s})$ and $c' = (i_1', i_2', \mu_1', \mu_2', \vec{s}')$ for which the coupling has to be found. The indices of the mass points coupling with a contact are paired with the corresponding coupling quantity by defining:

$$P((i_1, i_2, \mu_1, \mu_2, \vec{s})) :=$$
$$\{(i_1, 1-\mu_1), (i_1+1, \mu_1), (i_2, 1-\mu_2), (i_2+1, \mu_2)\}$$

Using this definition, the coupling $G_{c,c'}$ between c and c' is given by

$$G_{c,c'} := \sum_{\substack{(i,\mu)\in P(c) \\ (j,\nu)\in P(c')}} \mu \cdot \nu \cdot G_{i,j}.$$

Comparison with (7) and (9) reveals that the coupling between two contacts is given by

$$\Delta\vec{x}_{c'} = G_{c,c'} \cdot \vec{C}_c$$

Given $M \in \mathbb{N}$ contacts $\{c_i\}$, $i = 1 \ldots M$, the corresponding contact forces will be denoted by $\vec{C}_{c_i}$. The relative movement of contacts before any contact forces have been applied will be denoted by $\Delta\vec{x}_{0,c_i}$. These movements originate from other external forces and internal spring forces. The total relative movement of contacts after contact forces have been applied will be denoted by $\Delta\vec{x}_{c_i}$. The contact force vector is defined by

$$\vec{C}_{\{c_i\}} := \left(\vec{C}_{c_1}^T, \vec{C}_{c_2}^T, \ldots\right)^T. \tag{13}$$

A vector for the relative movements before contact forces have been applied is defined by:

$$\Delta\vec{x}_{0,\{c_i\}} := \left(\Delta\vec{x}_{0,c_1}^T, \Delta\vec{x}_{0,c_2}^T, \ldots\right)^T,$$

and also a vector for the relative movements after the contact forces have been applied:

$$\Delta\vec{x}_{\{c_i\}} := \left(\Delta\vec{x}_{c_1}^T, \Delta\vec{x}_{c_2}^T, \ldots\right)^T.$$

With matrix Q given by

$$Q := \begin{pmatrix} G_{c_1,c_1} & G_{c_2,c_1} & \ldots \\ G_{c_1,c_2} & G_{c_2,c_2} & \ldots \\ \ldots & \ldots & \ldots \end{pmatrix}$$

the relation between these quantities is given by

$$\Delta\vec{x}_{\{c_i\}} = Q \cdot \vec{C}_{\{c_i\}} + \Delta\vec{x}_{0,\{c_i\}} \tag{14}$$

This relation has to be solved respecting the condition given in previous section 3.6.2. It can be stated as a linear complementary problem (LCP) [Cat05] or as a nonlinear complementary problem (NCP) [SHNE10]. The LCP problem has often been stated for rigid body contact handling and is commonly solved with the projected Gauss Seidel algorithm (PGS). [SHNE10] solves the NCP problem by applying a Fletcher-Reeves type nonlinear nonsmooth conjugate gradient (NNCG) type method. Both methods greatly benefit from warmstarting where the result of the last update step is used as an initial guess for the current update step.

© The Eurographics Association 2011.

Both PGS and NNCG have been tested for the problem and NNCG has been found to output better results in most cases.

Because the time available for an update interval is limited, the time requirements have to be limited. The stopping criteria for the iterative solver has been chosen to a fixed numbers of iterations. This has the advantage of making the time consumption of the solver predictable, reducing variation in time consumption between update steps.

4. Results

All experiments have been done on an Intel(R) Core(TM) i7 CPU with 3.07Ghz. Ubuntu 11.04 has been used as the operating system.

Both approaches are stable even with a very high spring constant k_i and only one simulation update is done per frame ($\Delta t \approx 30ms$ for a simulation running at $30Hz$). For the simplified approach because of the approximation error described in figure 1 a high k_i leads to the thread behaving like a rigid object. A force applied to one end of the thread gets almost equally distributed among all mass points leading to a homogeneous movement of the whole thread. This behavior is illustrated in the supplementary material. This effect is reduced by decreasing Δt. The balancing between these values has been done by visual inspection and testing. A timestep of $\Delta t = 5ms$ has been found to provide good visual results with a stiff thread. Additionally the effect can be reduced by setting $\lambda < 1$. For $\lambda < 0.5$ the simulation gets unstable. A value of $\lambda = 0.5$ has therefore been chosen.

While for the normal approach a larger timestep still provides good visual results for the dynamic behavior of the thread, a large timestep increases the risk of a collision being missed and the thread tunneling through itself or other threads. For better comparison of the approaches a timestep of $\Delta t = 5ms$ has also been chosen for the normal approach. The dynamic behaviour can be viewed in the video added in the supplementary material which can also be downloaded at `https://www.ziti.uni-heidelberg.de/icm/de/forschung/microsim.html`.

The contact constraint solver is set to a fixed numbers of 50 iterations per integration update.

4.1. Integrator

As mention in section 3.3 the implicit integration scheme can be used with any integrator fulfilling (1).

An integrator storing the impulses of the mass points would apply an impulse to contact points. This results in the contacts being resolved and not re-detected in the next frame which causes instabilities. This problem is addressed in [SJTM08] with the predictor-corrector approach. Unfortunately in case of a tight knot involving big forces, a large time step can lead to the thread tunneling through itself. The low mass integrator avoids this problem by not remembering the impulse. It solves the differential equation

$$\dot{\vec{x}} = \mu \vec{F}$$

where μ is called the mobility. The $\mu\vec{F}$ models friction in a viscous fluid. The term $m\ddot{\vec{x}}$ is missing in this equation. If μ is big in relation to the mass m, the term $m\ddot{\vec{x}}$ is small in comparison to the other terms and can be omitted.

4.2. Tying a knot

The visual dynamic behavior and knot tying behavior of the simulation is tested on the reef knot, which is a common knot used to tie together two threads or ropes of equal diameter. Figure 3 shows a loose configuration of the reef knot. This is the starting point for the test simulation. The reef knot involves two threads, which are color coded in figure 3. To better visualize how the threads are simulated as discrete segments, adjacent segments are colored differently leading to two colors per thread.

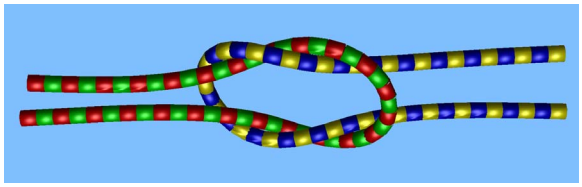

Figure 3: *Start configuration for knot tying simulation.*

To tighten the knot, the first mass point of each thread is fixed using the method described in section 3.5. The fixed mass point of the red-green thread is then pulled to the left putting the knot under stress. The knot fast converges to a equilibrium situation holding the threads together. The situation is shown in figure 4.

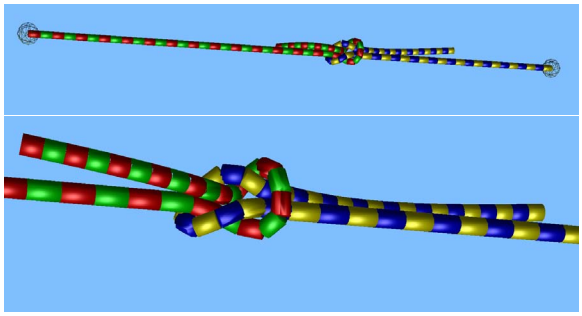

Figure 4: *Tying the knot.*

Gravity is added to the system, pulling the threads downwards. The new configuration is shown in figure 5.

© The Eurographics Association 2011.

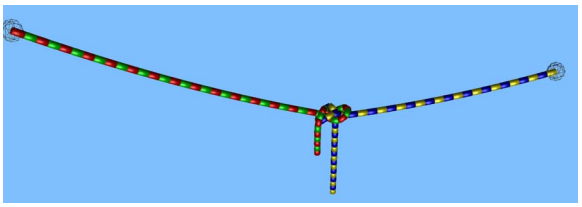

Figure 5: *Adding gravity.*

4.3. Contact constraint solver

The reef knot, as described in the last section, has been tied and contact forces have been computed using the contact constraint solver. This has been repeated over 90 integrations steps. After the constraints have been solved, the error in the computed forces has been measured by detecting how much the total forces at the contacts differ from a force fulfilling the criteria described in section 3.6.2. For the simulation to be stable, it is important that the maximal force error is small in every integration step. For this reason, the maximal force error of all contact forces has been measured after every integration step and the mean over all 90 integration steps has been calculated. This has been repeated with and without warmstarting for different number of iterations.

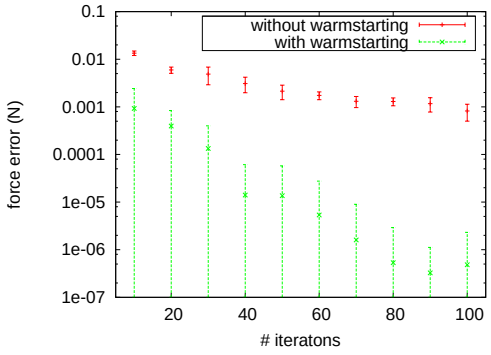

Figure 6: *Error in contact forces for different number of constraint solver iterations.*

The results for the described situation is shown in figure 6. These results vary depending on the configuration of the threads. For this reason the choice of 50 iterations has not been evaluated from figure 6 but by visual inspection from various situations. The value of 50 iterations has always yielded satisfying results when warmstarting has been enabled.

The results in figure 6 show an tremendous gain in precision by warmstarting. This effect is reduced when a more dynamic situation is analyzed. But the higher precision at resting situations is important because when the thread is at rest, errors in its movement are much more visible to the user. Without warmstarting the reef knot contains vibrating movements even when the number of iterations is set to 100.

4.4. Timings

The time complexity of both integration approaches are $O(n)$. Due to the more bands in the normal approach, it is expected to perform worse by a constant factor than the simplified approach. For the detection of collisions a spatial subdivision technique known as spatial hashing [THM*03] has been implemented which has a time complexity of $O(n)$. The complexity of the contact constraint solver is independent of the number of mass points n, but depends on the number of contacts c. The number of iterations is fixed at 50 and therefore does not contribute to the complexity. For every iteration, coupling between all pairs of contacts have to be calculated setting the complexity to $O(c^2)$.

Running time for different parts of the method are measured in the configuration described in the last section (the reef knots). The number of mass points of each thread is set to 50, 100 and 150 resulting in the total number of mass points in the simulation being set to 100, 200 and 300. The time needed for collision detections and both integration approaches is taken. The results are shown in table 1.

N	Collision detection (μs)	Integration (μs)	simplified Integration (μs)
100	126 ± 15	54 ± 8	14 ± 3
200	184 ± 22	110 ± 20	27 ± 5
300	265 ± 24	180 ± 40	40 ± 9

Table 1: *Timing for different number of nodes per threads when tying a reef knot. The number of nodes is the sum of nodes for both threads. N denotes the number of nodes.*

The time needed for resolving contact constraints is not a function of the number of mass points but of the number of constraints and therefore not included in table 1. It is also expected to be higher for the normal approach than for the simplified approach because calculating the coupling between contacts involves a matrix multiplication while for the simplified approach only a scalar multiplication is needed. Figure 7 shows the time the contact constraint solver took in dependence on the number of contacts. During the simulation of the reef knot in the last section, the number of contacts always stayed below 25. The number of contact constraint solving iterations is set to 50.

5. Discussion and Outlook

A novel method to simulate a stiff thread based on implicit integration has been introduced. Because implicit integration is unconditionally stable, the method allows the update interval to be high, reducing the computational requirements of the simulation. The integration method allows the time step

© The Eurographics Association 2011.

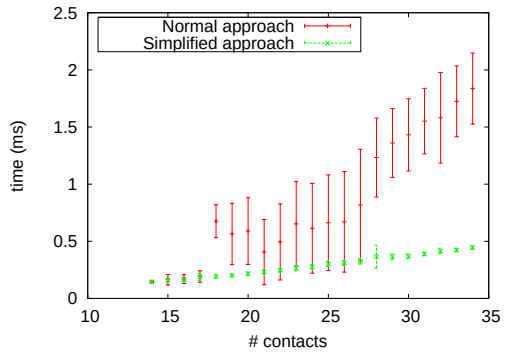

Figure 7: *Time for 50 iterations of contact resolution.*

to be arbitrarily large. The computational demands of an implicit integration scheme are handled by breaking the problem into two steps, the contact resulution and the main integration. The main integration step thereby becomes much simpler to solve and the contact resolution is a small problem by itself. A simplification for the integration which, as can be seen in table 1, is faster by about a factor of five, has been pointed out in section 3.4. Unfortunately, with the simplified approach large time steps in combination with a stiff thread lead to the thread behaving like a rigid object due to the approximation made during the integration.

In addition with a too large time step the risk of tunneling increases. While this could be solved with continuous collision detection, a time step of $5ms$ has been found not to stress the computation time to much while not posing problems with tunneling. The normal approach has a tendency of being more unstable when tightening knots. These instabilities can be reduced by increasing ε in (6). While this makes the behavior of the normal approach more similar to the behavior of the simplified approach, an ε can be found making the simulation stable while still allowing the thread to be very stiff.

The main bottleneck of the simulation is the contact constraint handling. Figure 7 shows, that for more than 35 constraints, the time needed for constraint solving gets dangerously close to $2ms$. But for simulation of the reef knot the number of constraints does not get above 25 keeping the time needed for constraint solving below $1ms$. While the reef knot is already a complicated knot, care must be taken if more complex knot are tied or several knots exist at the same time.

The time needed for contact resolution is quadratic based on the number of contacts. It would therefore be beneficial to divide the contacts into subsets of different knots assuming the knots do not influence each other much. Also tied knots could be identified as such and be simulated as rigid objects coupling with the thread requiring no contact constraints to be solved.

6. Acknowledgement

This work is kindly supported by the German Bundesministerium für Wirtschaft und Technologie (BMWi) under grant ZIM (KF2351202SS9). The method has been developed in cooperation with VRmagic GmbH in Mannheim.

References

[BLM04] BROWN J., LATOMBE J.-C., MONTGOMERY K.: Real-time knot-tying simulation. *Vis. Comput. 20* (May 2004), 165–179. 2

[BMcLS01] BROWN J., MONTGOMERY K., CLAUDE LATOMBE J., STEPHANIDES M.: A microsurgery simulation system. In *In Medical Image Computing and Computer-Assisted Interventions (MICCAI)* (2001), pp. 137–144. 2

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. 2

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete Elastic Rods. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (aug 2008), 63:1–63:12. 2

[Cat05] CATTO E.: Iterative dynamics with temporal coherence. *Game Developer Conference* (2005). 2, 5

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

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

[DSB99] DESBRUN M., SCHRÖDER P., BARR A.: Interactive animation of structured deformable objects. In *Proceedings of the 1999 conference on Graphics interface '99* (San Francisco, CA, USA, 1999), Morgan Kaufmann Publishers Inc., pp. 1–8. 2

[GPL08] GOYAL S., PERKINS N. C., LEE C. L.: Non-linear dynamic intertwining of rods with self-contact. *International Journal of Non-Linear Mechanics 43*, 1 (Jan. 2008), 65–73. 2

[HMC01] HILDE L., MESEURE P., CHAILLOU C.: A fast implicit integration method for solving dynamic equations of movement. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2001), VRST '01, ACM, pp. 71–76. 2

[Kla96] KLAPPER I.: Biological applications of the dynamics of twisted elastic rods. *Journal of Computational Physics 125* (1996), 325–337. 2

[LWL10] LARSSON K., WALLGREN G., LARSON M. G.: Interactive Simulation of a Continuum Mechanics based Torsional Thread. In *VRIPHYS* (Copenhagen, Denmark, 2010), Erleben K., Bender J., Teschner M., (Eds.), Eurographics Association, pp. 49–58. 2

[NMK*06] NEALEN A., MUELLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum 25*, 4 (Dec. 2006), 809–836. 1

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

© The Eurographics Association 2011.

[PK11] PUNAK S., KURENOV S.: Simplified Cosserat rod for interactive suture modeling. *Studies in health technology and informatics 163* (2011), 466–472. 2

[SHNE10] SILCOWITZ-HANSEN M., NIEBE S., ERLEBEN K.: A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *The Visual Computer 26* (2010), 893–901. 10.1007/s00371-010-0502-6. 2, 3, 5

[SJTM08] SPILLMANN, JONAS, TESCHNER, MATTHIAS: An Adaptive Contact Model for the Robust Simulation of Knots. *Computer Graphics Forum 27*, 2 (Apr. 2008), 497–506. 6

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

[TGAB08] THEETTEN A., GRISONI L., ANDRIOT C., BARSKY B.: Geometrically exact dynamic splines. *Comput. Aided Des. 40* (January 2008), 35–48. 2

[THM*03] TESCHNER M., HEIDELBERGER B., MÃIJLLER M., POMERANTES D., GROSS M. H.: Optimized spatial hashing for collision detection of deformable objects. In *VMV'03* (2003), pp. 47–54. 7

[Tho79] THORSON J.: Gaussian elimination on a banded matrix. Stanford Exploration Project, 1979. 3

Appendix A: Proof for Lemma 1

Proof: (12) is proved first by induction over i. For $i < n$ the right hand side in (5) is 0, so for $i = 1$ this means:

$$\begin{aligned} & D_1\vec{\delta}_{n,1} + O_1^+\vec{\delta}_{n,2} = 0 \\ \Rightarrow \quad & \vec{\delta}_{n,1} = -D_1^{-1}O_1^+\vec{\delta}_{n,2} = M_1^+\vec{\delta}_{n,2} \end{aligned}$$

Doing the induction step from $i \to i+1$ we again know from (5)

$$\begin{aligned} & D_i\vec{\delta}_{n,i} + O_i^+\vec{\delta}_{n,i+1} + O_i^-\vec{\delta}_{n,i-1} = 0 \\ \Rightarrow \quad & D_i\vec{\delta}_{n,i} + O_i^+\vec{\delta}_{n,i+1} + O_i^- M_{i-1}^+\vec{\delta}_{n,i-1} = 0 \\ \Rightarrow \quad & \vec{\delta}_{n,i} = -\left[D_i + O_i^- M_{i-1}^+\right]^{-1} O_i^+\vec{\delta}_{n,i+1} = M_i^+\vec{\delta}_{n,i+1} \end{aligned}$$

where the induction hypotheses has been used in the second line. The proof for (11) works analogous.

For (10) we start again from (5), where the right hand side now is $b_i\Delta\vec{E}_n$:

$$\begin{aligned} & D_n\vec{\delta}_{n,n} + O_n^+\vec{\delta}_{n,n+1} + O_n^-\vec{\delta}_{n,n-1} = b_n\Delta\vec{E}_n \\ \Rightarrow \quad & D_n\vec{\delta}_{n,n} + O_n^+ M_{n+1}^-\vec{\delta}_{n,n} + O_n^- M_{n-1}^+\vec{\delta}_{n,n} = b_n\Delta\vec{E}_n \\ \Rightarrow \quad & \vec{\delta}_{n,n} = \left[D_n + O_n^+ M_{n+1}^- + O_n^- M_{n-1}^+\right]^{-1} b_n\Delta\vec{E}_n \qquad \square \end{aligned}$$

© The Eurographics Association 2011.