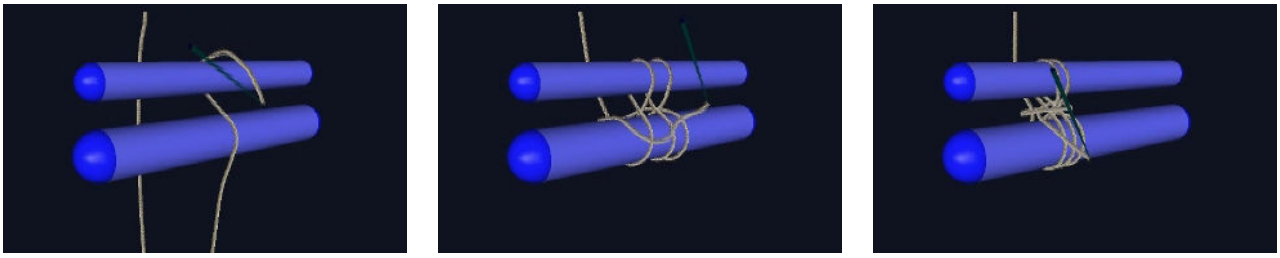
A Robust Method for Real-Time Thread Simulation

Blazej Kubiak *
University of Łódź

Nico Pietroni †
Fabio Ganovelli‡
Visual Computing Lab Isti cnr Pisa
Endocas Center for Computer assisted surgery

Marco Fratarcangeli§
Universitá La Sapienza Roma
Institute of Technology Linkoping



Abstract

In this paper, we present a physically based model for real-time simulation of thread dynamics. Our model captures all the relevant aspects of the physics of the thread, including quasi-zero elasticity, bending, torsion and self-collision, and it provides output forces for the haptic feedback. The physical properties are modeled in terms of constraints that are iteratively satisfied while the numerical integration is carried out through a Verlet scheme. This approach leads to an unconditionally stable, controllable and computationally light simulation [Müller et al. 2007]. Our results demonstrate the effectiveness of our model, showing the interaction of the thread with other objects in real time and the creation of complex knots.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: physically based animation, surgical simulation

1 Introduction

The main use of a thread simulator is certainly in endoscopic surgical simulation. Handling the surgical thread to make knots (which is required in many surgical procedures) is one of the most difficult tasks for a surgeon for it requires ambidexterity with the endoscopic forceps.
Simulating a thread is a intriguing problem because, although it seems simpler than for more complex 3D structures, the interaction with a thread involves some worst case situations for self-collision detection and contact handling, which are both fundamental to make knots. Furthermore, the thread is almost inextensible, and from the point of view of the simulation, this means that the methods modeling elasticity and using explicit time-integration schemes are poorly conditioned.

We implemented a method for simulating surgical thread based on Position Based Dynamics [Müller et al. 2007], modeling stiffness, bending, torsion and providing feedback for the haptic device. The collision detection is carried out by spatial hashing. We speed up collision detection introducing a hierarchical test on the curvature of the thread to early discard those portion of the thread that cannot self intersect. The rest of the paper proceeds as follows: in Section 2 we briefly review the state of the art in thread simulation; in Section 3 we describe the detail of our implementation and results and conclusion a reported in Section 6.

*e-mail: blazejkubiak@gmail.com
†e-mail:nico.pietroni@isti.cnr.it
‡fabio.ganovelli@isti.cnr.it
§e-mail:frat@dis.uniroma1.it

2 Previous Work

Most of the approaches to thread simulation are energy-based (i.e., define a system energy and derive it for computing the forces), and model the thread as a one dimensional chain of mass points, where adjacent mass points are connected by springs. If the spring are pure elastic elements following Hooke law, some care needs to be taken to avoid oscillation and instability of the simulation.

In [Phillips et al. 2002] this is done by making number of points adaptive with respect to the inter-points distance and using a modified integration scheme.

In [Wang et al. 2006] the relation between points includes also bending and torsion. A dissipation factor is also considered in the form of friction forces consequent to contact.

A continuous model is proposed in [Lenoir et al. 2002], where the thread is modeled as a spline and energy of the system if defined with an integral over the spline. Although the mass is distributed along the thread and not lumped at the points, the energy term is ultimately computed by discretizing the computation of elongation and bending of the curve.

A non energy based model is presented in [Brown et al. 2004] under with name of Follow The Leader (FTL) algorithm. It consists of first moving the nodes constrained by external action (e.g., grasped by a tool), and then heuristically moving the others trying to preserve the original inter-points distance. Although the absence of physical simulation is apparent, this method allows making complex nodes at interactive frame rate.

A critical aspect of all the methods is how self-collision is detected and how contact is handled. A common approach is to use a hierarchy of bounding spheres. For the simplicity of the thread structure and for the low number of points generally used (few hundreds), it turns out to be an effective solution, although the sphere is the worst fitting geometric primitive for a segment. As for more general simulation models, the collision generates impulse forces trying to

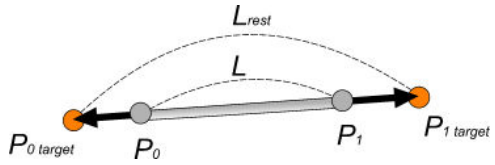


Figure 1: Stiffness constraint: The 2 particles p_0 and p_1 are displaced in opposite directions along the edge in order to restore the original length of the thread L_{rest}.

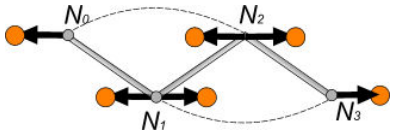


Figure 2: Bending constraint: A stiffness constraint is added between node N_i and node N_{i+2} in order to oppose to bending operations.

restore non contact situation and the friction due to continuous contact is modeled as a dissipative force.

3 Our Approach.

We model the physics of the thread using Position Based Dynamics [Müller et al. 2007]. This approach consists of modeling the physics as a set of constraints and then running the simulation iterating three steps: 1) moving the mass points according to their velocity and external action (e.g. grasping) 2) moving the mass points to satisfy the constraints 3) performing time integration. If the direction of movement is along the gradient of the constraint function then the linear and angular momentum are preserved and no ghost forces are introduced. This characteristic coupled with a Verlet scheme (which does not store velocity) guarantees a unconditionally stable method (please refer to [Müller et al. 2007] for further details).
Position Based Dynamics is particularly useful to handle collision and contact, where with energy based method stability is hard to achieve, and this is the main reason for our choice.

Stiffness and bending

We use a *Distance Constraint* $C(p_0, p_1) = \|p_1 - p_0\| - L_{rest}$, both to keep the inter-points distance constant (see Figure 1) and to model resistance to bending by constraining the distance between every second mass point as proposed in [Provot 1995] (see Figure 2).

Contact constraint

Self contact constraints are added whenever the cable self-intersects, i.e. when the distance between 2 non adjacent edges is less than the radius of the thread r. The constraint is of the form $C(p) = [p-(p_{n0}+p_v)]$ where $p_v = (\|p_{n0}-p_{n1}\|-r)\cdot\frac{p_{n0}-p_{n1}}{\|p_{n0}-p_{n1}\|}$ is the penetration vector and p_{n0} is the current position of point p.

Friction constraint

If a segment composing the thread is in contact, then its movement should be limited by adding friction. In the original approach, friction is modeled by manipulating the velocities. Instead, we chose to integrate the friction in the constraints projection. The reason is that the projection of the other constraints could bring the particles in a non-contact state *before* the velocities are updated and the risk of oscillating between contact state and non-contact state is greater.

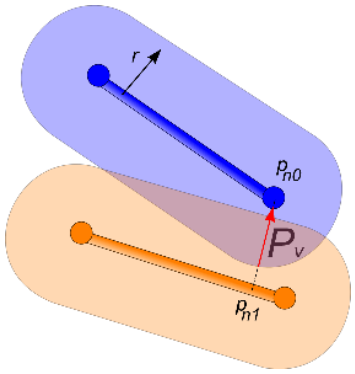


Figure 3: Collision constraint: Given the section of the thread r the cable self-intersect if the distance between 2 non adjacent edges is less than r. The penetration vector P_v define how to move the colliding segments in order to resolve the collision

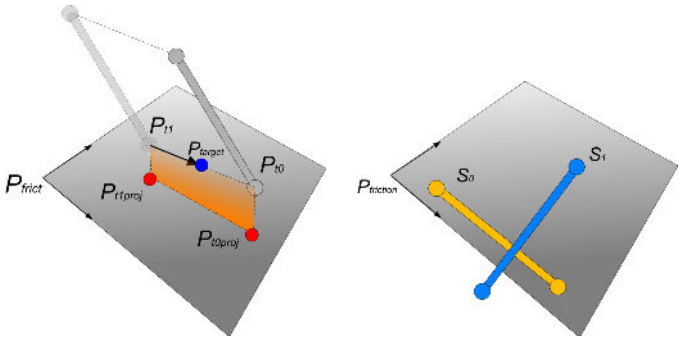


Figure 4: Left) Example of friction. right) Friction plane definition in the case of 2 segments in contact: the friction plane is defined as the best plane fitting the 2 segments.

Let $\Delta_i p$ be the displacement computed by the projection of constraint C_i, the point p be in contact with a surface locally approximate by a plane P_{fric}. Then the displacement is modified as:

$$\Delta'_i p = \Delta'_i p \cdot \mu P v_{frict}$$

, where μ is the friction constant and Pv_{frict} is the penetration vector. In other words we apply the Coulomb friction in the constraint formulation, i.e. replacing forces with displacement. If the contact involves the thread and a surface, then the friction plane P_{fric} is defined by the normal over the nearest point of surface, while if the contact involve two segments then the friction plane is defined as the plane passing through the 2 segments (figure 4.2). The choice of integrating the friction in the constraint resolution is an heuristic that maybe make more difficult to tune the parameter but from our experiments it leads to a more stable simulation.

Torsion constraint
To add a torsion constraint we first have to evaluate the torsion angle. While stiffness, bending and contact constraint can be easily defined over a one dimensional thread, the torsion cannot be derived solely on the position of the mass points. For this reason we add to the system a *material* vector for each segment. This vector is orthogonal to the segment and it is meant to have fixed orientation in material coordinates.
Without loss of generality, let us assume that at the beginning of the simulation the thread is straight and the torsion is 0 everywhere, so that all the material vectors have the same orientation.
For each segment we define the torsion angle as the angle between

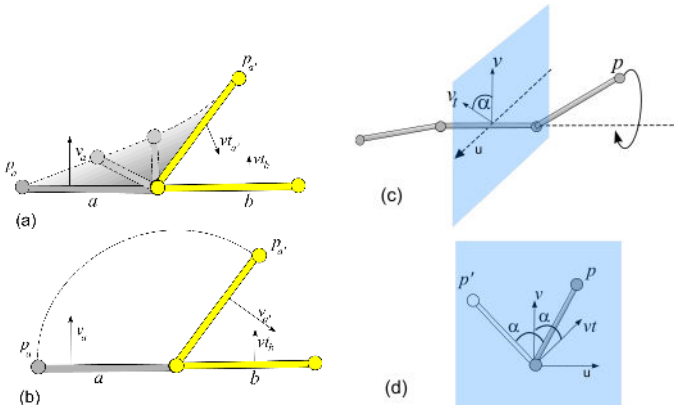


Figure 5: (a) computation of the material vector (b) finding the material vector under the assumption that there is no torsion (c-d) definition of the torsion constraint.

the actual material vector and a material vector computed assuming no torsion in the thread.

Figure 5.a shows two adjacent segments where the node p is moved from position p_a to the position $p_{a'}$ keeping the segment b fixed. The movement is done so that a torsion is created therefore the vector $vt_{a'}$ is not in the same plane as vt_b. Figure 5.b shows the same final configuration but this time the movement is a single rotation and no torsion is created, therefore $v_{a'}$ is in the same plane as vt_b. In other words the vector $v_{a'}$ is the value the material vector would have if the configuration was reached without applying a torque to the thread.
Figure 5.c shows a configuration of the thread where the two material vectors have been computed. Consider the projection of the node p onto the plane orthogonal to the central segment. If we rotate p by the torsion angle α the torsion would be 0, therefore we express the constraint as $C(p) = |\alpha|$.

Material vector computation
The material vector of each segment known at the beginning and it is updated step by step. Let us consider the segment s_i with material vector v_i at time t and $t+1$. We find the rototranslation RT_i as $\min\{RT|RTs_{it} - s_{it+1}||\}$ and update the material vector as $v_{it+1} = RT_i v_{it}$. This approximation relies on frame to frame coherency and on the fact that the thread has a very stiff behavior, so that the segments have almost constant length.

4 Collision Detection

Because the segments of the thread are equally sized we used a regular partition of the space and the Spatial Hashing technique with temporal marks introduced in [Teschner et al. 2003]. We also exploit the one dimensional nature of the model to define an quick rejection test based on the discrete curvature of the thread. The idea is to consider the angles between consecutive segments of the piecewise rectilinear curve and to check if such angles would allow a self-intersection of the curve.

Let us consider Figure 6.(a) representing a thread made of four segments where two ends have been joined to form a non selfintersecting (i.e. simple) polygon. The angles indicated with α_i are the *turns* taken to walk the polygon clockwise.
In general it holds $\|\sum_{i=0}^{i<n} \alpha_i\| = 2\pi$ with α_i signed (positive clockwise, negative counterclockwise). If the polygon is convex, than $\|\sum_{i=0}^{i<n} \alpha_i\| = \sum_{i=0}^{i<n} \|\alpha_i\|$ while if it is non convex $\|\sum_{i=0}^{i<n} \alpha_i\| < \sum_{i=0}^{i<n} \|\alpha_i\|$.

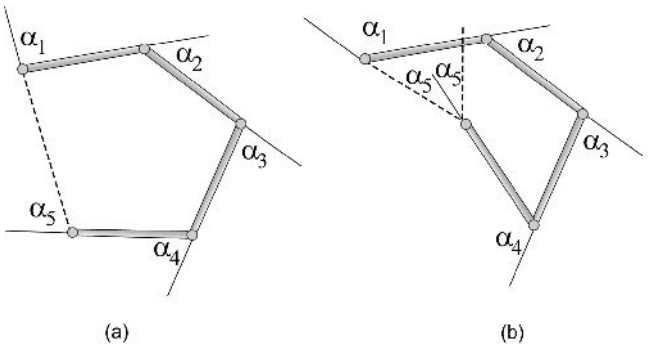


Figure 6: Simple pruning test for self collision detection based on angles: if $\sum_{i=0}^{i<n} \|\alpha_i\| <= 2\pi$, then self-intersection can not occur.

We use the fact that if a polygon is not simple then it cannot be convex to derive a rejection test for self intersection, i.e. if we can prove the polygon is convex that it cannot be non simple.

Since the thread is embedded in *3D*, we would need to find a plane on which the projection of the thread (plus the closing segment) is a convex polygon, which would become too much demanding for a quick rejection test. Instead we simply compute the *3D* angles between consecutive segments $\overline{\alpha_i}$ and test if $\sum_{i=0}^{i<n} \|\overline{\alpha_i}\| <= 2\pi$. If this inequality holds, the thread is guaranteed not to self-intersect. The intuitive meaning of this test is that if we flatten the thread on to a plane preserving all the angles and so that they have the same sign, the thread will not self intersect because it simply is not *bent enough* to self intersect. We use this test in a hierarchical fashion, starting from the whole thread and recurring on failure to reject the self-intersection.

5 Visual and Haptic feedback

In order to have a smooth shape of the rope, we refine it at rendering time using the *Chaikin's Algorithm* [Chaikin 1974], starting from the set of particles used during the simulation. We render each segment of the rope using textured *impostors*, in order to minimize the number geometric primitives. We draw a set of connected polygons that were transformed respect to the eye point of the scene using a cylindrical symmetry. The overload due to this rendering technique is negligible and it has been successfully used to produce the final visual examples shown on videos. The framework for knot tying simulation include a 6 degree haptic device to allow real-time knot tying, the **FREEDOM6S** interface [J.Demers 1998]. The user interact by grasping and freely moving a grasped node using the haptic device as shown in the first part of the video. The resulting *feedback force* is computed by considering as linear springs the 2 segments adjacent to the grasped node while the frame rate of haptic device is guaranteed by performing a linear extrapolation in a dedicated *thread*.

6 Conclusions and future work

We applied our model to simulate real time knot tying. Figures 7, 8 and 9 show some examples of interactive knot tying. The simulation runs at 70fps on a Intel Pentium 4 - 3,0 GHz using 90

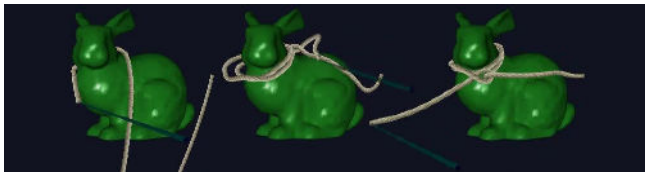


Figure 7: Strangling the bunny

Figure 8: Example of a knot with 2 ropes.

particles and 320 sampled points for ropes rendering.
We implemented the technique using the VCG library for the geometry concepts [VisualComputingLab 2005b], IDOLib for simulation primitives (time integration and contact)[VisualComputingLab 2005a] and the Haptik Library [de Pascale and Prattichizzo 2007] for interface with generic haptic hardware.
The video attached to this paper was produced grabbing the graphics window during an interactive simulation using the haptic device described in 5.

With respect to the *Follow-The-Leader* algorithm proposed in [Brown et al. 2004] that only takes into account stiffness and assumes the thread has no mass, our solution also considers gravity and torsion. Thanks to the stability and controllability of position based dynamics, our algorithm can model a very stiff thread, that is more similar to a real surgical thread with respect to the explicit mass spring formulation exposed in [Wang et al. 2005].

For future research, we plan to formulate new constraints on the thread in order to simulate the suturing process. In this case, it may be rather simple to constrain the cable to slide trough holes. Another direction of research is to investigate the possibility of adapting sampling of the thread and adaptive time step of integration. This was not necessary in our current setting but it will be when using longer threads without losing the detail of the representation.

References

BROWN, J., LATOMBE, J.-C., AND MONTGOMERY, K. 2004. Real-time knot-tying simulation. *The Visual Computer 20*, 2-3, 165–179.

CHAIKIN, G. 1974. An algorithm for high speed curve generation. *Computer Graphics and Image Processing 3*, 346–349.

DE PASCALE, M., AND PRATTICHIZZO, D. 2007. The haptick library: A component based architecture for uniform access to haptic devices. *IEEE Robotics and Automation Magazine, (to appear)*.

J.DEMERS, J.BOELEN, I. 1998. Freedom 6s force feedback hand controller mpb technologies inc. *IFAC Workshop on Space Robotics.*.

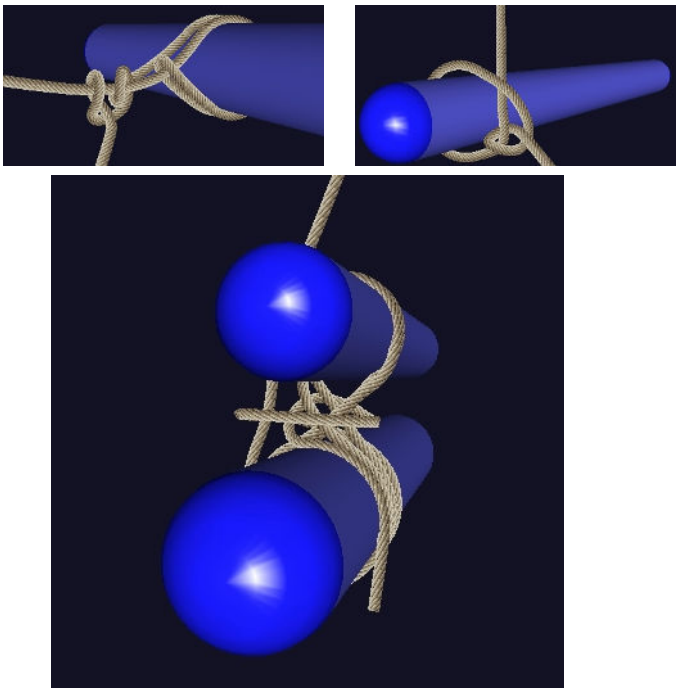


Figure 9: Detail views on other knots.

LENOIR, J., MESEURE, P., GRISONI, L., AND CHAILLOU, C. 2002. Surgical thread simulation. Modelling and Simulation for Computer-aided Medecine and Surgery (MS4CMS), vol. 12, 102–107.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2, 109–118.

PHILLIPS, J., LADD, A. M., AND KAVRAKI, L. E. 2002. Simulated knot tying. In *ICRA*, IEEE, 841–846.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, 147–154. ISBN 0-9695338-4-5.

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of the Conference on Vision, Modeling and Visualization 2003 (VMV-03)*, Aka GmbH, Berlin, T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, Eds., 47–54.

VISUALCOMPUTINGLAB. 2005. Interactive deformable objects library. more info on: http://idolib.sf.net.

VISUALCOMPUTINGLAB. 2005. Visualization and computer graphics library. more info on: http://vcg.sf.net.

WANG, F., BURDET, E., DHANIK, A., POSTON, T., AND TEO, C. L. 2005. Dynamic thread for real-time knot-tying. In *WHC '05: Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, IEEE Computer Society, Washington, DC, USA, 507–508.

WANG, F., BURDET, E., VUILLEMIN, R., AND H.BLEULER. 2006. Knot-tying with visual and force feedback for vr laparoscopic training. IEEE Computer Society, 5778–5781.