芸術科学会論文誌 Vol.4 No.4 pp.159 – 167

Pre-Step and Post-Step Deformation Schemes for Fastening Band-Shaped Twisted Cloth

Norio Sato, Kazuhiro Suzuki, Qinglian Guo, and Kyoko Kato
Dept. of Information and Computer Science
Kanazawa Institute of Technology

Abstract

We have developed a simulation program that presents the process of fastening band-shaped cloth objects. These objects are twisted in 3D space to mimic the movements of tying garments such as Kimono sash. Even using extensive collision handling, this simulation is tough. The difficulty lies in the complex object contacts: The movements of different parts of a long contiguous deformable object conflict with each other causing numerous collisions during the fastening process; Such collisions cause implausible deformations; Moreover, the collisions in different angles are prone to fatal penetrations that are not simple to remove. We propose three schemes to solve these problems. One is a pre-step mechanical scheme for calculating fastening force to move the object smoothly. Other two are a post-step geometrical correction scheme to smooth deformations and a set of schemes to remove penetrations. These schemes proved to enable robust and visually realistic simulations with low cost.

1 Introduction

Cloth modeling and simulation (or animation) is a matured topic in computer graphics (CG). In this paper, we present how to simulate the process of fastening a "twisted cloth object" that we refer to as "3D ribbon", on which no work has been reported. We have encountered with a problem of much heavier contacts among deformable objects, compared with previous researches on cloth simulation such as draping on still-life objects, or fitting to a body, or hanging in the air.

Figure 1 shows a snapshot how our prototyped software works. The initial user's input is a 2D curve, which is high-lighted. We generate a 3D ribbon as rendered with texture attached. Then, we fasten it around some other object as rendered in wireframe.

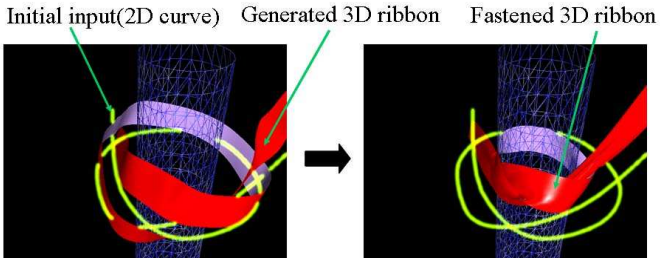


Figure 1. A snapshot: how the prototyped software works.

In the rest of this paper, section 2 explains the difference of our problem from those of related works. Section 3 introduces our prototyped software, and offers the pointers to the Section 4 and Section 5 that describe our algorithms in detail. Section 6 shows results and evaluates our achievement. Section 7 briefly discusses the extensions to make the achievement useful.

2 Related Works

Knot theory [1], where knots are defined as twisted closed loops in 3D space,is not directly relevant to our work. Two knots are considered equivalent, if one of them can be deformed to the other. Although the physical concepts such as width and forces are ignored, the theory gives us some hints. One useful property for our work is that twisted curves on a 2D plane can be regarded as the projection of "alternate knots" in 3D space.

Cloth modeling and simulation (or animation) is directly relevant to our research. In most solutions in literature[2, 3, 4, 5, 6, 7, 8, 9, 10], the cloth is modeled as a mass-spring network [2, 3] or sometimes a chain of rods [5], which makes it possible to simulate its deformations such as forming wrinkles [2, 3, 4, 5, 6], fitting 2D patterns to human bodies or sawing them[2]. Visual reality and simulation robustness are the essential requirements for this kind of simulation.

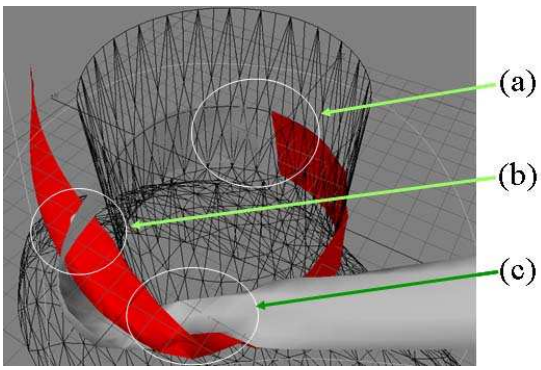


Figure 2. Movement and touches to fasten a ribbon: (a)surface to surface, (b)edge to surface, and (c)edge to edge.

Sharing the solutions with these previous works such as collision handling, a new problem must be solved in our simulation. As shown in figure 2, we must move a 3D "ribbon", which is a twisted contiguous deformable object, smoothly through the free space that is sandwiched by itself and the still-life object and is getting narrower and narrower while we tighten the ribbon. Such a movement causes conflicts in different parts of the ribbon. These parts collide with each other in different angles, not always "surface to surface", but also "edge to surface", and "edge to edge".

3 Overall Features

3.1 Prototyped Software

Figure 3 shows the construction of our prototyped software. The components are shown in boxes and described in detail in section 4 and section 5. The input and output of each component is shown in Figure 1, and has already been explained in Section 1.

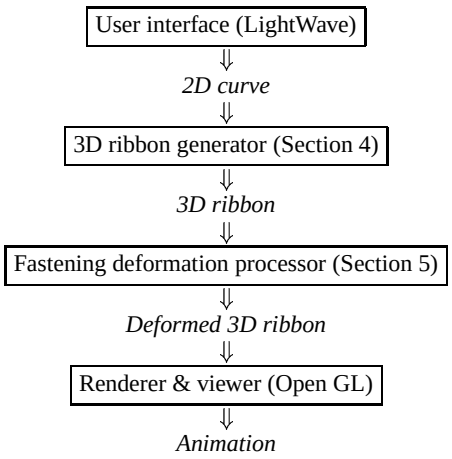


Figure 3. Prototyped software construction.

The software is coded in VC++(R) 6.0 using OpenGL library. We have integrated it to a popular commercial modeling software LightWave 3D (R) (referred to as LW) as a plug-in [11] to share its facilities.

3.2 How to generate a 3D ribbon ?

Since it is difficult for a user to input a curve in 3D space when using a regular 2D input device such as a mouse, we provide with a function to generate a 3D ribbon from a 2D curve that is defined by the user freely by using a mouse. The 3D ribbon is generated by adding a displacement at each crossing point along the third dimensional axis.

We smooth the generated 3D curve to define the initial state of a ribbon. Then, we add width in orthogonal direction with the curving to define a ribbon. The rest of the problem is reduced to the cloth modeling. We populate the mesh vertices with masses and connect them by virtual springs.

3.3 How to fasten a ribbon ?

Fastening a ribbon in real world is a subtle feedback process that is controled by human hands. In our simulation, the fastening force is simple and reproducible so that we may show the process in a comprehensive way. Instead of two (invisible) hands, we impose forces to every part of the ribbon prior to the calculation of deformations, which we refer to as “pre-step fastening force calculation”.

3.4 How to deform a ribbon ?

The “mass-spring model” is promising. The resistance force of springs to the elongation and shrinking deforms the model properly. “Collision handling” is a big issue. Repulsions to free space cause another effect than the spring resistances, and thereby, affect the deformations of the object so that it may not intersect with itself or with other objects. Because the collision test is an expensive computation, we have implemented the “hierarchical object subdivision” scheme to minimize the number of the tests.

Two side effects occur as a consequence of the responses to collisions. One is the over- and under-elongations of springs that no longer justify the spring model. The other is the penetrations between objects (we refer to as “errors”). Because both problems stem from the discretized model and time step, they are very difficult to circumvent by any calculation accuracy.

One remedial but effective solution is geometric corrections being done each time after the collision handling. Two iterative processes we have implemented solve these problems. One is for smoothing the deformation, which we refer to as “post-step correction”. This increases the deformation stability significantly, though causes errors. The other is for recovering from errors, which we refer to as “error recovery”. Both corrections play essential roles in our simulation where objects contact heavily with each other.

4 3D Ribbon Generator

4.1 Defining a 3D Curve that Represents the Initial State of Ribbon

We regard a twisted 2D curve as a “regular projection” of a string “alternate knot” in 3D space: A crossing point is regarded as two distinct points in 3D space. Walking along the string from one end to the other, the crossing over and beneath the crossing points come alternately[1].

The 3D curve is recognized as a sequence of points that are connected by straight lines. One point corresponding the crossing is “pulled” along the specified axis (typically “y-axis”), whereas the other is “pushed” so that they can make space for the specified width of the ribbon. Then, we draw a smooth 3D curve based on the LW standard curve called “Catmall-Rom Spline Curve”, which is a parametric three-degree curve using four adjacent points[12].

4.2 Adding width to 3D curve to create 3D ribbon mesh data

As illustrated in figure 4 upper, for each plotted point, we find a vector to add width that is orthogonal with the tangent to the curved plane on which the 3D curve runs. The lower depicts the result.

Roughly speaking, the vecotr at point P_i is the average of $\overrightarrow{P_{i-2}P_{i-1}} \times \overrightarrow{P_{i-1}P_i}$, $\overrightarrow{P_{i-1}P_i} \times \overrightarrow{P_iP_{i+1}}$, and $\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_{i+1}P_{i+2}}$, where “$\times$” means the cross product of two vectors[1] We add the specified width along with this vector to both sides of point P_i, and allocate the specified number of particles on this width line.

5 Fastening Deformation Processor

Figure 5 shows the pseudo-code of the ribbon generator. The details are described in the following subsections.

[1]At point P_0, this vector is fixed along the axis (typically “y-axis”) for the space making, and at P_n the average of the formar two.

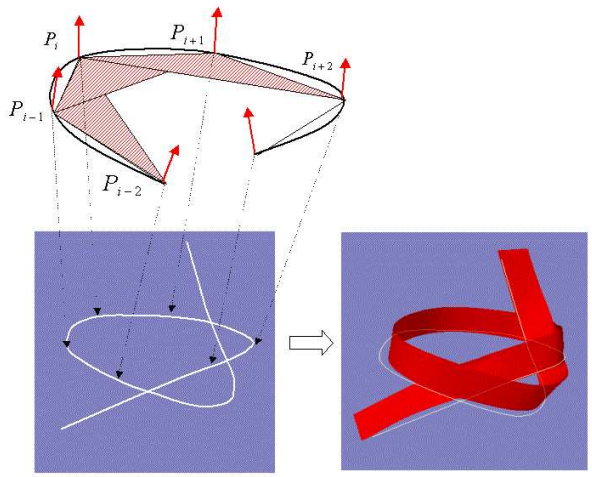


Figure 4. Adding width to 3D curve to create 3D ribbon mesh data: how to define the ribbon surface and an example.

Setting material properties (see Sec. 5.1)
For each time step {
  Calculating pre-step fastening force (see Sec.5.2)
  Calculating acceleration, velocity,
    and displacement (see Sec.5.3)
  Detecting collisions (see Sec. 5.4)
    while ∃ collision
      Update velocity, displacement
  Smoothing by post-step correction (see Sec.5.5)
    while ∃ over-stretched edge
      Apply post-step correction
  Recovering from errors (see Sec.5.6)
    while ∃ edge penetration
      Apply error recovery schemes
}

Figure 5. Conceptual pseudo-code of the fastening deformation processor.

5.1 Setting Material Properties

We generate a loosely knotted ribbon automatically as described in the previous section. This can be input into our simulation system via LW API. We model the cloth as a "mass-spring" network. The springs that connect adjacent particles give the resistance to *stretching* and *shrinking*, those along with the diagonal lines give the resistance to *shearing*, and those that connect the particles of distance two give the resistance to *bending* (refer to [2]).

LW allows us to design a background object, which can be input by the LW API as a set of polygon meshes. Every its polygon has its fixed normal vector, and we give the vertices boundless mass ($m^{-1} = 0$, where m is the particle mass).

5.2 Calculating Pre-Step Fastening Force

As shown in figure 6, we populate forces at every part of the ribbon. We move every particle of the ribbon toward one of its ends along the tangent direction of the smoothed curve of the ribbon. The force strength is made proportional to the distance from the center axis of the background object to the particle[2], and thereby, we smooth the movement of the ribbon.

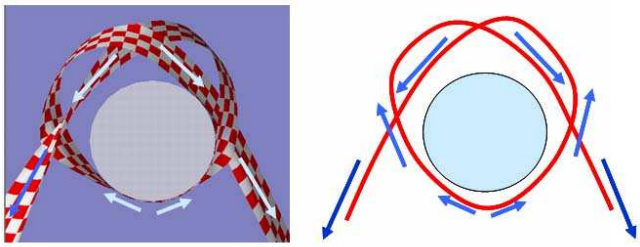


Figure 6. Fastening pre-step force calculation.

5.3 Calculating Acceleration, Velocity and Displacement

For each particle, we calculate from the spring resistance the acceleration, the velocity, and the displacement for the movement to deform the ribbon. We add air resistance to disperse energy and stabilize the simulation, while gravity and frictions are excluded to fasten the ribbon smoothly. We use "Runge-Kutta fourth order method" for the mechanical equation to minimize numeric errors.

5.4 Detecting Collisions and Adding Repulsions

5.4.1 Testing Collisions

A collision test is a proximity test between two triangles that do not share any vertex. Two types of collisions are essential to handle. One is *particle to polygon*, and the other is *edge to edge* for simulation and visual coherency reasons, respectively. For each vertex of one triangle, its proximity to the other triangle is tested. For each edge of one triangle, the proximity to the other triangle is tested.

Our test scheme is highly preventive from penetrations. To do so, we use a "time bounding box" for each polygon, which surrounds its "trajectory" from the current position to the assumed position for the next step. For each pair of polygons, if their time bounding boxes intersect with each other, their precise intersection is tested.

For testing "particle to polygon" type collisions, as illustrated in figure 7, we test the intersection between the trajectory line of the particle from the current (D) to the next step position (D')and two triangles that are the current ($\triangle ABC$) and the next positions of a polygon ($\triangle A'B'C'$).

For testing "edge to edge" type collisions, as illustrated in Figure 8, we use a rectangular trajectory of the edge from its current and next positions. Dividing the rectangular into two triangles, the test

[2]For the particle at P_i for $i > n/2$ along the length of the ribbon, the fastening force f_i is $(\sum_{j=i-m}^{i+m}(P_j - P_{j-1})/\sum_{j=i-m}^{i+m}|P_j - P_{j-1}|) \times l_i$, where m is some number (we use the number of particles of width), and l_i is some value propotional to the distance from the center axis to the particle (we use such a value as to move the particle by an edge length during one time interval). The direction is opposit, i.e., $-f_i$ for $i < n/2$.

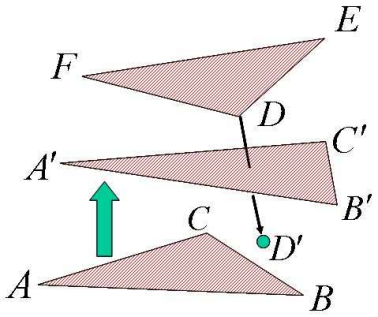


Figure 7. Testing a particle-polygon type collision.

is reduced to several intersection tests between a line and a triangle (see [13] for the formulation).

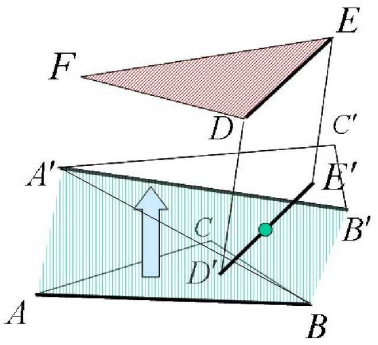


Figure 8. Testing an edge-edge type collision.

5.4.2 Adding Repulsions

Each time a particle is influenced by a collision, a new velocity is given according to the momentum conservation law (see literature[8] for the formulation). We use a small value for the repulsion coefficient for simulation stability. For a particle-polygon collision, as in figure 7, the new velocity is calculated by "dot product" of $\overrightarrow{DP}$ and the normal vector of $\triangle ABC$ (see [8] for the formulation). For an edge-edge collision, as in figure 8, the repulsion of the edge DE is determined by the "cross product" of $\overrightarrow{DE}$ and $\overrightarrow{AB}$.

5.4.3 Reducing Redundant Collision Tests

Reducing redundant collision tests is mandatory. While no library is available, we appreciate the thorough prescriptions in literature[2, 8] to implement two schemes from scratch and to evaluate their effect:

Space enumeration scheme[8]: The number of space elements increases by somewhere between linear to square order of magnitude for the number of polygons.The efficiency is quite insufficient for our simulation. A coarse ribbon of even 4×300 mass-spring granularity is heavy enough to stop the fastening simulation on its way.

Hierarchical object subdivision scheme[2]: Although this scheme is a much bigger task to implement than the above, its optimizing effect is much more significant. The number of tests increases only by logarithm order of magnitude for the number of polygons. The performance gain proved to be significant, which allows us to simulate fine-grained ribbons of more than 8×300 mass-spring in a few minutes.

In the latter scheme, we build a *hierarchical tree* for each object. Each leaf represents a polygon. Each node represents either a region of leaves or a group of regions. Taking one polygon, three adjacent polygons such that their contour length is minimum, are put together to form a group of one rank higher in this tree.

For each node, a "bounding box" and a "curvature" are calculated efficiently in bottom-up order by "space OR" and "bitwise AND" operations, respectively. Self and external collisions can be detected in top-down order for the hierarchical tree, where the curvatures are used for the collision test possibility between adjacent regions, while the bounding boxes are used for the test between non-adjacent regions.

One simplification we have made is to build such trees only once before the simulation starts. For simple shaped ribbons, building the tree for each simulation step is relatively expensive. As described later, this scheme is very useful also to process the "error detection" efficiently.

5.5 Smoothing by Post-Step Correction

The mass-spring is a fairly reasonable model for cloths, only while their elongation or shrinking is within a small range. However, due to the discretized time and model itself, the collision responses inevitably cause over elongation or shrinking. This is fatal to our simulation.

Using non-linear resistance of springs leads us to numeric instability due to the granularity of the time step. One safe way is, as shown in figure 9, to moderate the deformation by means of geometric modification. This is used by a simulation of hanging flags[9]. When two particles are over-stretched, they are brought together along the axis while preserving their center. Doing this correction iteratively until over and under stretches disappear[3], we can circumvent implausible deformations. A serious drawback to our simulation is that such a geometric correction, as this correction, is done independently of the mechanical calculation and collision handling, causes too many penetrations into the ribbon itself and into the background object.

To minimize this undesirable side effect, we have improved this correction scheme so that the positional adjustments take place along the tangent direction to the "smoothed" surface curvature. We approximate this displacement, as shown in figure 9, by moving the particles along the length direction of the ribbon and then along the width direction. Doing this bit by bit iteratively by ten times as its limit until the over and under elongations disappear, it proved that the side effect decreases significantly. We observe that the ribbon no longer penetrates to the background object, leaving a few self-penetrations around crossing parts of the ribbon.

However, even if we can decrease the penetrations, this side effect is unavoidable[4] , and therefore, should be removed by the error

[3] within 10% of the natural length

[4] A recent proposal[10] on smoothing, which is applied to draping and folding, is to use "adaptive time step"(to handle multiple collisions) and to "adjust velocities instead of postions" (to adjust the spring stiffness), and thereby to guarantee penetration free simulation. In the problem of fastening a ribbon, however, we would have to decrease the time step constantly and would have little chance to increase it to proceed the simulation.

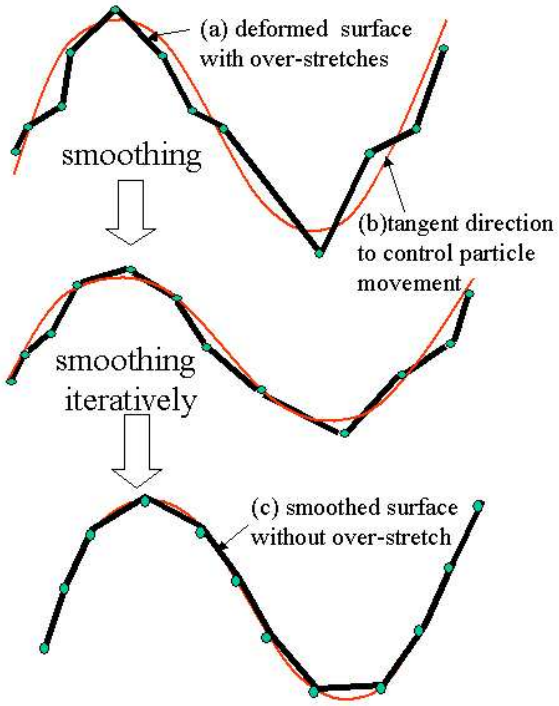


Figure 9. Smoothing by post-step (geometric) correction.

recovery schemes as described in the next section.

5.6 Recovering from Errors by Post-Step Penetration Removal

5.6.1 Causes of Errors

In numeric integration, since the time step is some discrete value, erroneous surface penetrations are by any means unavoidable. In our simulation, there are two causes: 1) As shown in Figure 10, One collision response (at P1) may cause another collision (at P2) before the next time step. 2) As shown in figure 11, The above mentioned post-step correction, in our experience, more often causes penetrations. Unless we remove the penetrations, further steps of the simulation result in serious incoherences sooner or later before we complete the fastening.

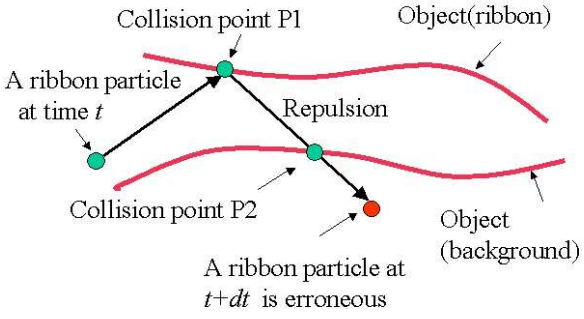


Figure 10. Errors due to collision response.

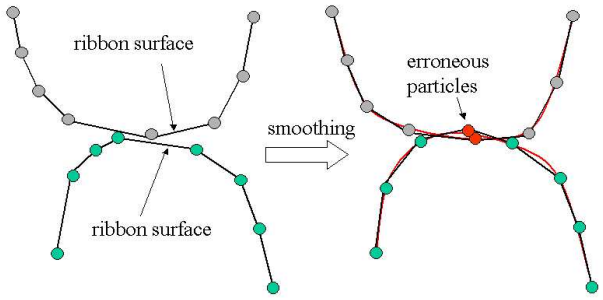


Figure 11. Errors due to smoothing.

5.6.2 Detecting Errors

The algorithm to detect “errors” is almost the same as the collision detection test. This is closely related to the hierarchical object division scheme. By using curvature and bounding boxes set up in the hierarchical tree, the “penetrating edges”can be found efficiently.

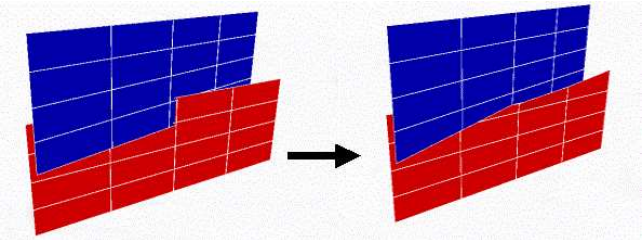
5.6.3 Error Patterns and Recovery Schemes

The penetrations of the ribbon into the background object is simple, and can be recovered by the following scheme:

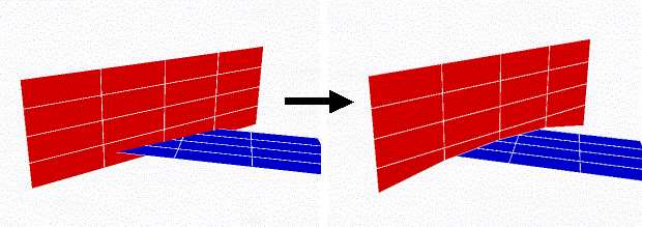
(Scheme A) Recovery from “ribbon penetration into background (still-life) object”: This is applied to all the erroneous edges of the ribbon. By counting the number of the neighborhood edges to an edge that is not erroneous, we decide which particle of an edge is erroneous and is to be moved. The appropriate moving direction is the normal vector of the triangle of the background object running “from the inside to the outside of the object”. The moving length is the distance from the particle to the crossed triangle.

The self-penetrations of the ribbon, which is specific to our problem, are much more complicated. The ribbon has no inside or outside. Some of the parts collide in different angles with each other. Figure 12 classifies typical error patterns that are caused by the self-collision patterns of a ribbon in three different angles. We approximate the collisions in different angles as one of these three cases. We remove the errors of these three error patterns by the following two schemes:

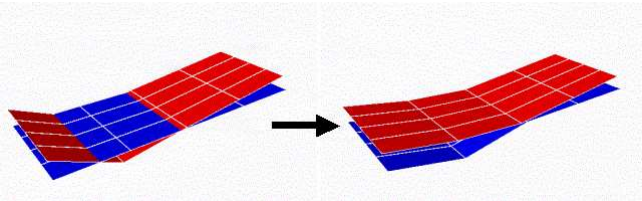
(Scheme B) Recovery from “ribbon-edge penetration into ribbon-edge or into ribbon-surface”: This scheme is applied only to the erroneous edges along the “ribbon edge”. Such an edge has two erroneous particles. As shown upper two pictures in figure 12, we “shrink and curl the ribbon edge along the width direction”. To do this, as shown in figure 13, we determine the moving direction by taking the average of the tangent vectors along the curvature starting from the erroneous particle to, if present, the first folding particle as on the right-hand side, otherwise to the other edge of the ribbon as on the left-hand side. For edge vectors $e_0, e_1, \ldots, e_{n-1}$, if $e_0 \cdot e_i < 0$ for some i, the starting particle of the e_i is the folding particle. The average of $e_0, \ldots e_{i-1}$ are the direction to move the particle. Along the width direction, several neighborhood particles are moved together, as shown in figure 14. By applying this also to several neighborhood particles “along the ribbon edge”, we shape a smooth wrinkle.



Ribbon-edge to ribbon-edge error recovery
by scheme B.



Ribbon-edge to ribbon-surface error recovery
by scheme B.



Ribbon-surface to ribbon-surface error recovery
by scheme C.

Figure 12. Self-penetration error patterns and recoveries.

(Scheme C) Recovery from "ribbon-surface penetration into ribbon-surface": This is applied to all the erroneous edges of the ribbon. The difference from Scheme A is that the normal vector shows us only the angle, not the direction. We take the one side of the plane containing the crossing triangle where the majority of the neighborhood particles exist, as the correct side to move. This is known by the "dot product" of two vectors: One is the normal vector of the crossed surface; The other is from the crossing point to the particle position.

5.6.4 Iterating Error Recoveries

Figure 15 shows the error recovery process as a whole. Since it is difficult to "recognize the error patterns" in figure 12 and the former two patterns are more likely, we prioritize the schemes. The above mentioned "scheme A" is independent of others, and so can precede or follow other schemes. Scheme B (for ribbon-edge) should precede scheme C (for others). For each iterative step, therefore, we apply the schemes A, B and C in this order. The error detection processing precedes each execution of these schemes.

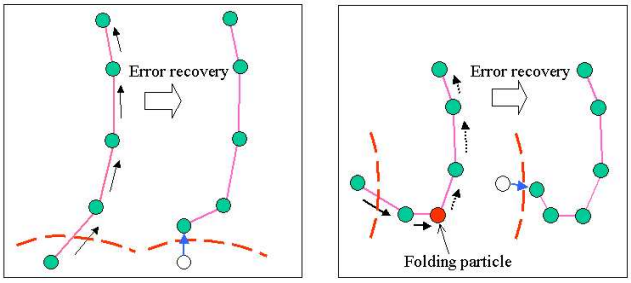


Figure 13. Moving errorneous particles on the ribbon-edge in Scheme B.

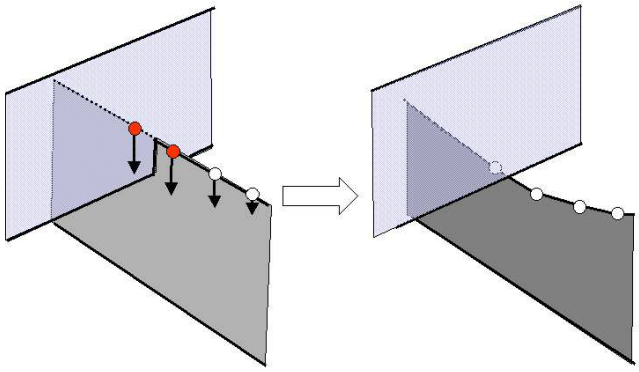


Figure 14. Penetration removal in Scheme B (ribbon-edge to ribbon-surface or to ribbon-edge).

We iterate the error recovery processes until all the penetrations disappear, or no further improvement is obtained. This physically means an approximation of "multiple collision" handling (e.g., queuing the possible collisions) within a discrete step, which would not work for the interaction between deformable edges and surfaces.

```
while (∃ errors and ∃ improvement) do {
  Setup hierarchical tree;
  Test ribbon penetration into background object;
  while (∃ error) Apply "scheme A";
  Setup hierarchical tree ;
  Test ribbon self-penetration;
  while (∃ error along ribbon-edges )
    Apply "scheme B";
  Setup hierarchical tree ;
  Test ribbon self-penetration;
  while (∃ error) Apply "scheme C";
}
```

Figure 15. Conceptual pseudo-code of the error recovery processing.

6 Results and Evaluations

6.1 Visual Reality

Figure 16 and shows an example. The upper and middle shows the same simulation from different angles, where the parameters are: 16×400 particles, $M = 1.0, k1, k2, k3 = 5.0, dt = 0.01$, where M is the total mass of the ribbon, $k1$, $k2$ and $k3$ are spring resistance for metric, shearing and bending, dt is the time step. In the lower case, the granularity is lower: 8×400 particles, $M = 1.0, k1, k2, k3 = 10.0$, and $dt = 0.01$, i.e. the same stiffness as the upper.

Figure 17 shows another example of the same granularity and the stiffness as the upper and lower case of figure 16.

We observe that the fastening process has almost been completed, leaving few penetrations that are invisible. With higher granularity (16×400 particles), the material is more smooth having more realistic wrinkles than lower cases (8×400 particles). Figures 18 and 19 show other examples. A ribbon is fastened around a cylinder and a small object. In both cases, cloth-like wrinkles are observed.

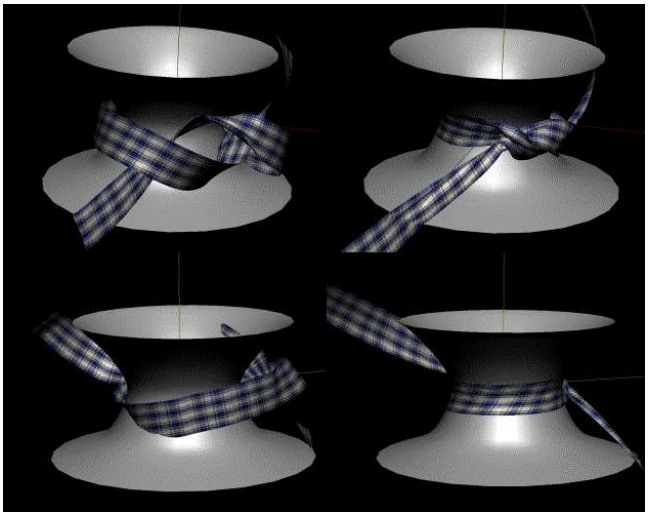


Figure 17. Fastening a narrower ribbon (granularity of 16×400 particles, other parameters are the same as Figure 16) around a hourglass cylinder .

6.2 Simulation Robustness

The simulation is stable for a wide range of the stiffness for $k/m = 0$ to 10^6, and $dt = 0.01$, where k is the spring registance, m is the mass of a particle, and dt is the time step value. In our experience, the appropriate values for cloths are several 10^4. The values of fewer orders of magnitude are suitable for "necklace chains", and those for larger orders of magnitude are for "papers".

6.3 Computational Cost

With the granularity of a few thousand particles (8×400), the simulation speed is reasonably fast, though a bit slower than real-time, within ten minutes in today's ordinary desktop and laptop computers that use 1 GHz to 2 GHz Pentium CPUs. Doubling the width granularity (16×400), it takes several tens of minutes to complete the fastening. In either case, providing with a viewing software, we can reproduce the simulation at high speed.

7 Future Extensions

Following extensions are to be considered:

Tightness: We can show to what degree the fastening is done during the simulation. One good measure could be the number of error recovery iterations. In our experience, the ribbon moves until it gets sandwiched by itself and the background object. Continuing further, the crossing gets twisted thin band-shape. When the ribbon gets tight, a recovery causes another recovery. As a consequence, the number of the error recovery iterations rapidly increases and the simulation gets slow.

Non-band-shaped ribbons: The 3D ribbon generation should be extended to cover the "ribbons in our daily life", such as scarfs and cravats, that have varying width. Some more flexible interface should be provided to allow the objects of arbitrary width. Adding width is restrictive at present. In the process of smoothing the 3D curve, too wide ribbons may cause penetrations to itself. When the penetration is not heavy, the "error recovery" works well (this happens even at present), otherwise the simulation fails. Therefore, non-flat surfaces should be generated to allow for wide ribbons. One approach could be to generate rounded, waved or folding surfaces. Another approach could be to build natural twisted surfaces with wrinkles around the background object at ribbon generation time, e.g., to increase the width step by step, while applying the error recovery schemes at each step.

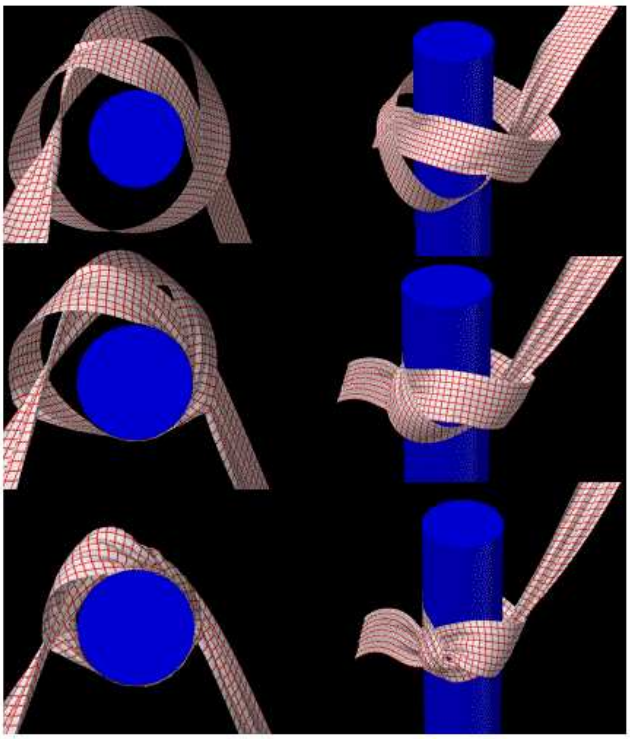


Figure 18. A process of fastening a ribbon (16×400 particles) around a cylinder.

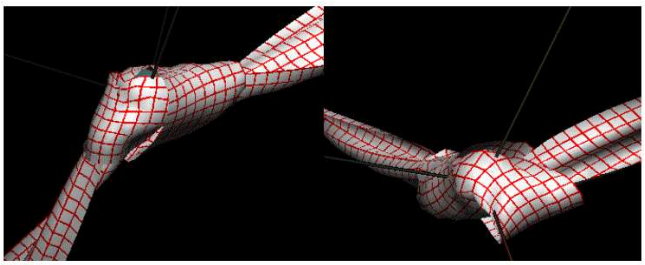


Figure 19. Fastening ribbons (8×400 and 16 particles) around a small object.

Complicated knot patterns: Most of the decorative garment knotting is non-alternate, and not simple. We need some function to draw free and smooth curves in 3D space. Next, more dynamic fastening forces should also be considered. It could be appripriate to combine simple knottings to shape compli-

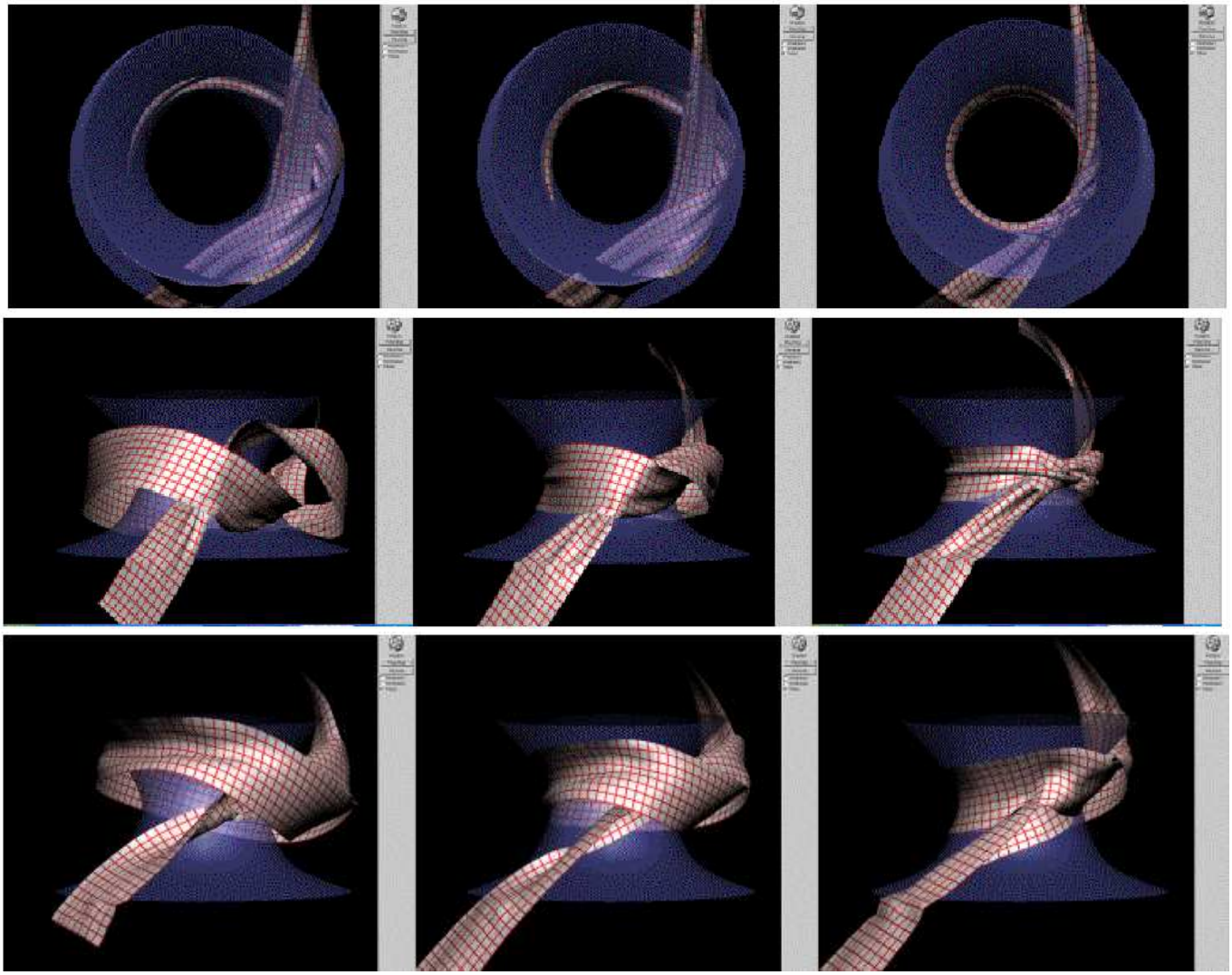


Figure 16. Simulation examples of fastening a ribbon around a hourglass cylinder. (Upper and middle: 16×400 **particles** $M = 1.0, k1, k2, k3 = 5.0, dt = 0.01$**, where** M **is the total mass of the ribbon,** $k1$**,** $k2$ **and** $k3$ **are spring resistance for metric, shearing and bending,** dt **is the time step. Lower:** 8×400 **particles,** $M = 1.0, k1, k2, k3 = 10.0$**, and** $dt = 0.01$**, i.e. the same stiffness as the upper.)**

cated knotting, e.g., to fasten the knots one by one, from inside to outside, as we do in real life.

8 Conclusion

Through empirical studies, we have succeeded in simulating the fastening process of a band-shaped cloth that is twisted in 3D space. Since existing techniques do not suffice in such a simulation that involves heavy touches among deformable objects, we have proposed three remedial but effective solutions:

- The pre-step fastening force, not just pulling the both ends of a ribbon, accelerates the speed and enables smooth movement in the simulation, but is left for further study together with the future direction as described in the previous section.
- The post-step smoothing itself is mandatory for stability in such a simulation as fastening a ribbon. We have shown that a fine-grained low-path filtering technique works with low cost.
- The post-step penetration removal is the core part. It enables us to skip the multiple collision handling in the physical calculation phase. Taking ribbon edge collision into account is the key to the fastening problem.

Our achievement is a good starting point to develop a useful software in the future that enables more complicated and decorative knotting. Such a software will be useful for those who would like to design or to learn how to tie cravats, how to bind sashes for dressing Kimono, etc.

9 References

[1] Mitsuyuki Ochiai, Shuji Yamada and Emiko Toyoda, *Computer Aided KnotTheory*, Makino Shoten (1996).

[2] Pascal Volino and Nadia Magnenat-Thalmann, *Virtual Clothing Theory and Practice*, Springer Verlag (1998)

[3] Edited by Donald H. House and David E. Breen, *Cloth Mod-*

eling and Animation, A K Peters, Ltd. (2000)

[4] Kunii, Gotohda, *Modeling and Animation of Garment Wrinkle Formation Processes*, Computer Animation proceedings, Springer-Verlag, pp 131-146, 1990

[5] Chiyi Cheng, Ying-Qing Xu, Jiaoying Shi, and Heung-Yeung Shum: *Physically Based Real-time Animation of Hangings*, CGI 2001.

[6] Pascal Volino, Nadia Magneat-Thalman: *Geometirc Wrinkles on Animated Surfaces*, WSSG proceedings, 1999

[7] Sunil Hadap, Endre Bangerter, Pascal Volino, and Nadia magnenat-Thalman: *Animating Wrinkles with Clothes*, pp.175-182 Proceedings of the conference on Visualization '99: (1999)

[8] Roberto Bigliani and Jeffrey W. Eischen, *Collision Detection in Cloth Simulation*, In Chapter 8 of Edited by Donald H. House and David E. Breen, *Cloth Modeling and Animation*, A K Peters, Ltd. (2000)

[9] Mathieu Desbrun, Mark Meyer, and Alan H. Bar, *Interactive Animation of Cloth-like Objects for Virtual Reality*, In Chapter 9 of Edited by Donald H. House and David E. Breen: *Cloth Modeling and Animation*, A K Peters, Ltd. (2000)

[10] Robert Bridson, Ronald Fedkiw, and John Anderson, *Robust Treatment of Collisions, Contact and Friction for Cloth Animation*, SIGRAPH 2002, ACM TOG 21, pp.594-603 (2002)

[11] D STORM Inc., *LightWave Plug-in Server Development*, http://www.dstorm.co.jp

[12] D STORM Inc., *How to Calculate Catmall-Rom Spline*, http://www.dstorm.co.jp

[13] Kevin Kaiser, *3 D Collision Detection*, In Section 4.5 of Edited by Mark DeLoura, *Game Programming Gems*, Born Digital Inc., (2000)