COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1431

SPECIAL ISSUE PAPER

Animating strings with twisting, tearing and flicking effects

Witawat Rungjiratananon[1]*, Yoshihiro Kanamori[2], Napaporn Metaaphanon[3], Yosuke Bando[4], Bing-Yu Chen[5] and Tomoyuki Nishita[1]

[1] Complexity Science and Engineering, The University of Tokyo, Kashiwa, Japan
[2] Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Japan
[3] Square Enix, Tokyo, Japan
[4] TOSHIBA Corporation, Tokyo, Japan
[5] National Taiwan University, Taipei, Taiwan

ABSTRACT

String-like objects in our daily lives, for example shoelaces, threads, rubber cords, plastic fiber and spaghetti, have a wide variety of materials. Such string-like objects also exhibit interesting behaviors such as twisting, tearing (by stretching or twisting), and bouncing back when pulled and released. In this paper, we present a method that enables these behaviors and simulates such materials in traditional string simulation methods that explicitly represent a string by particles and segments. Specifically, we offer the following three contributions. First, we introduce a method for handling twisting effects with both uniform and non-uniform torsional rigidities. Second, we propose a method for estimating the tension acting on inextensible strings in order to reproduce tearing and flicking (bouncing back), whereas the tension for an *extensible* object can be computed via stretched length. The length of an inextensible object is maintained constant in general, and thus, we need a novel approach. Third, we introduce an optimized grid-based collision detection for accelerating the computation. We demonstrate that our method can produce visually plausible animations of string-like objects with various material properties, and it is a fast framework for interactive applications such as games.

KEYWORDS
twisting, tearing, flicking, string animation

Supporting information may be found in the online version of this article.

***Correspondence**
Witawat Rungjiratananon, Complexity Science and Engineering, The University of Tokyo, Kashiwa, Japan.
E-mail: im_witawat@yahoo.com

1. INTRODUCTION

String-like deformable objects play an important role in representing hair strands, threads, elastic rods, cables, and ropes in computer graphics. For realistic animations of such objects, we have to reproduce their interesting behaviors such as bending, stretching, twisting, tearing, plasticity, and flicking when pulled and released, according to their material properties. For example, threads are made of yarn that barely stretches but easily tears. An elastic rod made of rubber can be twisted and flicked but is hard to break. A partially braided rope such as fourragère has a non-uniform torsional rigidity. Soft strings such as spaghetti and candy are easily cut by stretching and twisting. In the rest of this paper, we refer to such string-like objects as *strings*.

In order to simulate a string, several traditional methods have been proposed, such as mass-spring systems [1,2], rigid multibody serial chains [3], geometric approaches [4,5], and elastic energy-based models [6–8]. However, all behaviors (i.e., twisting, tearing, and flicking) of a string are not introduced together in a single framework.

The handling of inextensible strings, such as threads and cables, poses another technical challenge. To prevent excessive elongation of inextensible strings, many length-constraint schemes called *strain limiting* have been developed [9–13]. With strain limiting, however, the tearing simulation becomes difficult. Although an *extensible* string will break when its length or strain reaches a certain breaking point, we cannot see when an *inextensible* string will tear on the basis of the constrained length. Moreover, beside the fact that an inextensible string is not elongated

by their own weight, under a large applied force such as a large pulling force, the string should be elongated according to its material property. However, controlling material property becomes hard since the strain is unrelated to the applied force with strain limiting.

In this paper, we present a method that can handle twisting, tearing, and flicking of strings in real time. Our method is a simple pseudo-physically-based model which is easy to implement, yet visually plausible results can still be achieved. Our method is applicable to traditional simulation methods that explicitly represent a string by particles and segments. Our implementation is based on *chain shape matching* (CSM) [5], which is a simplified version of the more versatile deformation method, *lattice shape matching* [4], because CSM inherits and enhances several advantages of lattice shape matching (e.g., CSM is fast, easy to implement, and numerically stable). In this paper, we offer the following three contributions:

1. We introduce a simple method for twisting effects by adding twisting angles into each segment of a string, which can handle both uniform and non-uniform torsional rigidities.
2. We propose a method in estimating the tension of tearing and flicking effects in an inextensible string whose actual tensile stress and strain values are constrained from strain limiting. Furthermore, we consider the torsional tension in handling plasticity and tearing from twisting as well.
3. We introduce a collision searching scheme for efficient collision handling by using a grid-based data structure, which has less number of neighbors to be searched compared to typical searching schemes.

This paper is an extended version of [14]. Please refer to Supporting Information† for a summary of differences.

2. RELATED WORK

Simulation of twisting strings: Considerable research on the twisting effects in string simulation introduced various models for solving the Cosserat and Kirchhoff energy equations. Bertails *et al.* [6] introduced a mechanical model called *super helices* to simulate human hair on the basis of the Kirchhoff theory. However, handling collision responses is not straightforward because of the implicit representation of hair strands. Spillmann and Teschner [7] explicitly represented the centerline of an elastic string and used the finite element method (FEM) to solve the Cosserat energy equation. Recently, Bergou *et al.* [8] introduced a discrete model to simulate elastic strings on the basis of the Kirchhoff theory. However, the twisting angles are computed with a quasi-static assumption. Thus, the twisting of non-uniform torsional rigidity along the string is not addressed. There are also several works on pseudo-physical models that can capture the twisting effect without solving the energy equations. Hadap [3] introduced a model that captures the torsion effect by integrating a torsion spring into each joint of rigid links. However, strings cannot be stretched and collision handling is not straightforward because the motion is propagated from top to bottom in one single pass (not affected backward). Selle *et al.* [2] represented a hair strand by a chain of tetrahedrons of springs and captured the torsion effect by introducing appropriate altitude springs. However, the configuration of springs is complex, and auxiliary particles are required along a string.

Strain limiting for inextensible strings: In order to handle inextensible objects simulated by deformation models, a variety of methods for stretch resistance have been continuously proposed, from Provot's iterative post-processing edge constraint [9] to a more recent constraint method based on impulse [13]. Some alternative ways of stabilizing stiff simulation were also proposed [10–12]. These methods and many of their sequels have a common goal, to limit the maximal strain to a certain threshold. Accordingly, these kinds of methods are problematic in the case of excessive stretch or when rupture should occur. Metaaphanon *et al.* [15] proposed a method to deal with cloth tearing by using a mass-spring model. However, it tears cloth by checking lengths of springs; when and where yarns of cloth are cut were not directly related to user-applied external forces and cloth material properties but were dependent on how the method constrains the springs.ss

3. CHAIN SHAPE MATCHING

Before describing the details of our algorithms, this section first briefly introduces *Chain Shape Matching* (CSM) [5], the basis model used in this paper. In the CSM, a string is represented as a chain of particles connected by segments (see Figure 1 (a)). The particles are grouped into multiple overlapping *chain regions* with the *region half-width* $w_{CSM} \in \{1, 2, 3, \dots\}$. The chain region half-width corresponds to the stiffness of the string. The particles are independently moved by external forces, and then an optimal rigid transformation (i.e., rotation and translation) of each region is computed. The rigidly transformed positions of the particles are called *goal positions*. The goal position of each particle is weighed in the overlapping regions by *particle per-region mass*. Finally, each particle is updated toward the goal position.

4. TWISTING EFFECTS

Based on the CSM, a string in our model is represented as a chain of $(n + 1)$ particles connected by n segments (Figure 1(a)). A segment $i \in \{1, 2, \dots, n\}$ has a twisting angle θ_i tracking how much the segment is twisted. The twisting angle can be represented as an angle between

†Supporting information may be found in the online version of this article.

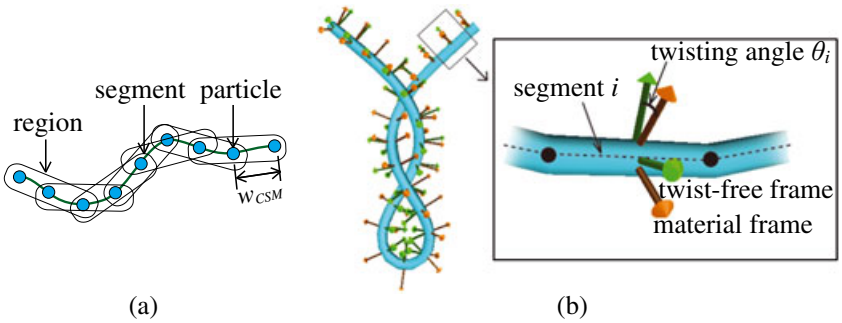


Figure 1. Our string model: (a) multiple overlapping chain regions in the chain shape matching; (b) the twisting angle of a segment of an elastic string is an angle between a twist-free frame and a material frame.

a *twist-free frame* (*bishop frame*) and a *material frame* (Figure 1(b)). In the initial state, we specify an initial angle θ_i^0 of each segment i according to the shape of the string. The twisting angle is assigned to each segment, not for each particle, to avoid the ambiguity.

The behavior of twisting can be clearly observed when a string, clamped at both ends, is twisted at one end. Therefore, we use this scenario for our explanation (Figure 2). When we twist one clamped end with an angle θ_t, the angle θ_i of the segment is increased. The increment of the twisting angle of the segment is propagated to the next segments in order to minimize the elastic energy in the string. In other words, the string tries to minimize the twisting angles between each connected segment. We compute and update a goal twisting angle for each segment, similarly, to finding a goal position for each particle in the shape matching.

First, we group the segments into multiple overlapping chain regions with the region half-width $w_{twist} \in \{1, 2, 3, \dots\}$, which affects the propagation speed of the twisting angles in the string or the torsional rigidity; the larger the w_{twist} is, the faster the change of twisting angles is propagated. The size of each region in a string can be varied in handling non-uniform torsional rigidity. The minimized twisting angle increment $\Delta\theta_k^{region}$ of each region k is computed by averaging the twisting angle increment $\Delta\theta_j = \theta_j - \theta_j^0$ of the segments in the region k weighted by mass m_j:

$$\Delta\theta_k^{region} = \frac{\sum_{j \in S_k} m_j \Delta\theta_j}{\sum_{j \in S_k} m_j} \tag{1}$$

where S_k is a set of segments within region k. Then, the θ_i of each segment i is updated with the twisting angle increment $\Delta\theta_i^{segment}$. The goal twisting angle increment $\Delta\theta_i^{segment}$ is calculated by summing the twisting angle increment $\Delta\theta_k^{region}$ of each region k to which segment i belongs to:

$$\Delta\theta_i^{segment} = \sum_{k \in \Re} \Delta\theta_k^{region} \tag{2}$$

$$\theta_i = \Delta\theta_i^{segment} \tag{3}$$

where $\Re$ is the set of regions to which segment i belongs to.

While a segment is updated to the goal twisting angle, a torque occurs in the cross-section, causing the change of rotational velocity $\omega_i^{segment}$ in the segment's tangential axis. In each time step Δt, a segment is rotated by the rotational velocity first, and then it is updated to the computed goal twisting angle in Equation (3). The rotational velocity

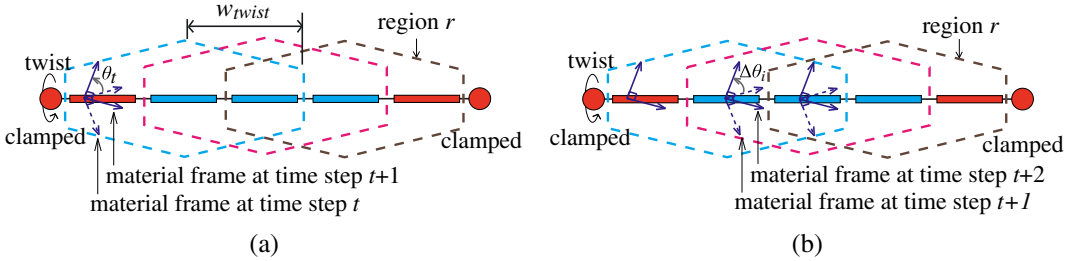
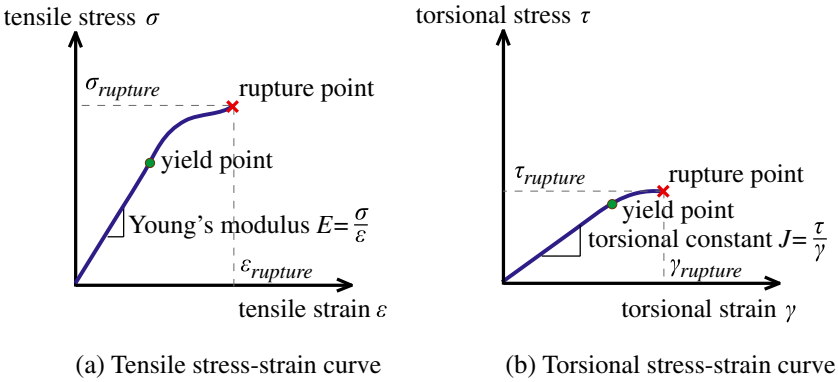


Figure 2. (a) An elastic string clamped at both ends is twisted at one end with a twisting angle θ_t. (b) The increment of the twisting angle is propagated to the next segments.



(a) Tensile stress-strain curve (b) Torsional stress-strain curve

Figure 3. Typical tensile and torsional stress–strain curves of a material: (a) tensile stress–strain curve; (b) torsional stress–strain curve.

and the time evolution of the twisting angle are computed as follows:

$$\omega_i^{segment} \leftarrow \omega_i + I\,\Delta\theta_i^{segment} \tag{4}$$

$$\theta_i \leftarrow \theta_i + \Delta t\,\omega_i^{segment} \tag{5}$$

where I_i is an inertia moment of segment i.

The twisting force $\mathbf{f}_i^{twist}$ can be treated as an external force exerted on particle i and derived from the elastic energy equation [8] as follows:

$$\mathbf{f}_i^{twist} = \frac{\beta}{L}(\theta_{i+1} - 2\theta_i + \theta_{i-1})\left(\frac{-\kappa\mathbf{b}_{i+1} - \kappa\mathbf{b}_{i-1}}{2l}\right) \tag{6}$$

$$\kappa\mathbf{b}_i = 2\frac{\mathbf{e}_{i-1} \times \mathbf{e}_i}{|\mathbf{e}_{i-1}||\mathbf{e}_i| + \mathbf{e}_{i-1} \cdot \mathbf{e}_i} \tag{7}$$

where $\kappa\mathbf{b}_i$ is the curvature binormal, $\mathbf{e}_i$ is the segment vector, l is the length of the segment, β is the twisting stiffness of the string, and L is the total length of the string.

5. TEARING AND FLICKING EFFECTS

This section briefly reviews the material science of a string and then describes *strain limiting*, a widely used technique used to constrain the length of an inextensible string from excessive elongation. Finally, we describe our method in handling tearing and flicking effects, which was previously difficult because of strain limiting.

5.1. Stress and Strain

In material science, the strength of a string is associated with its stress–strain curve [16]. There are many kinds of the stress–strain curves, depending on the direction of the force used in the material strength test, for example, tensile, compressive, shear, and torsional stress–strain curves. Because tearing is typically affected by tensile and torsional stresses, we consider only tensile and torsional stress–strain curves of the material in our model. The tensile stress–strain curve shows the relation between an average force per unit area of a cross-section surface and the elongation of a string. The torsional stress–strain curve shows the relation between an average torque on a cross-section surface and the twisting of a string. Examples of both curves are shown in Figure 3.

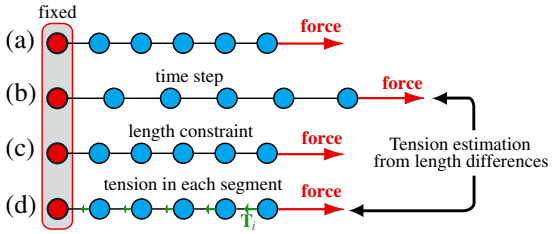


Figure 4. A simple example of the tension computation. From (a) to (c), an ordinary length-constrained string simulation is performed. In (d), tensions are estimated by calculating forces that make the particles move from their unconstrained positions to the constrained positions, yielding a result equivalent to (c).

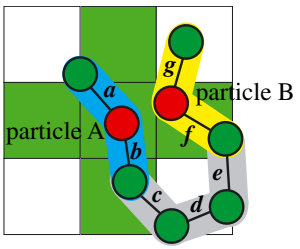


Figure 5. A two-dimensional illustration of our optimized searching scheme. When performing a collision detection between Particles A and B, segment collision tests between {*a*, *g*}, {*a*, *f*}, {*b*, *g*}, and {*b*, *f*} capsules are tested.

The tensile stress σ and torsional stress τ of a string are the average force and the average torque per unit area of a cross-section surface, respectively:

$$\sigma = \frac{\|\mathbf{F}_n\|}{A} \tag{8}$$

$$\tau = \frac{\|\mathbf{T}_n\|}{A} \tag{9}$$

where A is the cross-sectional area, $\mathbf{F}_n$ is the normal force, and $\mathbf{T}_n$ is the normal torque. The normal direction is the vector in the cross-section surface's normal direction.

The tensile strain ε and torsional strain γ of a string are expressed as the ratio of the elongation ΔL to the initial length L_0 and the change in twisting along the axis of the segment, respectively:

$$\varepsilon = \frac{\Delta L}{L_0} = \frac{L - L_0}{L_0} \tag{10}$$

$$\gamma = (\theta_{i-1} - \theta_{i+1})r \tag{11}$$

where L is the current length of the string and r is the radius of the segment. Note that $\Delta\theta_i = \theta_i - \theta_i^0$ is the change in the twisting angle from the rest state, not the tensional strain. The torsional strain comes from the difference of the twisting angles between connected segments, which are segments $i-1$ and $i+1$.

Along the curve, the material exhibits elastic behaviors until the *yield point*. Prior to the yield point, the material will return to its original shape if the applied force or torque is removed. The slopes of this elastic region are the *Young's modulus* $E = \sigma/\varepsilon$ and the *torsional constant* $J = \tau/\gamma$ in the tensile and torsional curves, respectively. Once the yield point is passed, the material becomes plastic; some fractions of the deformation will be permanent and irreversible. As deformation continues, the material will break when the stress or strain reaches the *rupture point*.

The stress–strain curve can be derived via the strength testing of a material sample, stored as a data set of the experimental result. The stress–strain curve of most materials in the elasticity state is linear, and thus, the part of the curve from the origin to the yield point can be stored as a constant value. Still, the data set is required for the curve in the plasticity state. In our implementation, we simply approximate the curve by a line with a constant slope that fits the curve best. As a result, our implementation uses two constant values to represent the stress–strain curve in the elasticity and plasticity states together with two constants for the yield point and rupture point.

5.2. Strain Limiting

By using traditional methods or CSM, strings are usually excessively stretched under its own weight and with large applied force. Instead of using the large Young's modulus, which can lead to numerical instability, position constraints are often imposed so that the length of each segment i does not exceed a certain threshold L_i^{max} [9–13], which is called *strain limiting*. Although our method of tension estimation described in the next subsection is applicable to any strain limiting approach, for our implementation, we used the position constraints method of Müller *et al.* [11].

By denoting the position vector of particle i by $\mathbf{x}_i$, we constrain the length $L_i = \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$ of segment i below

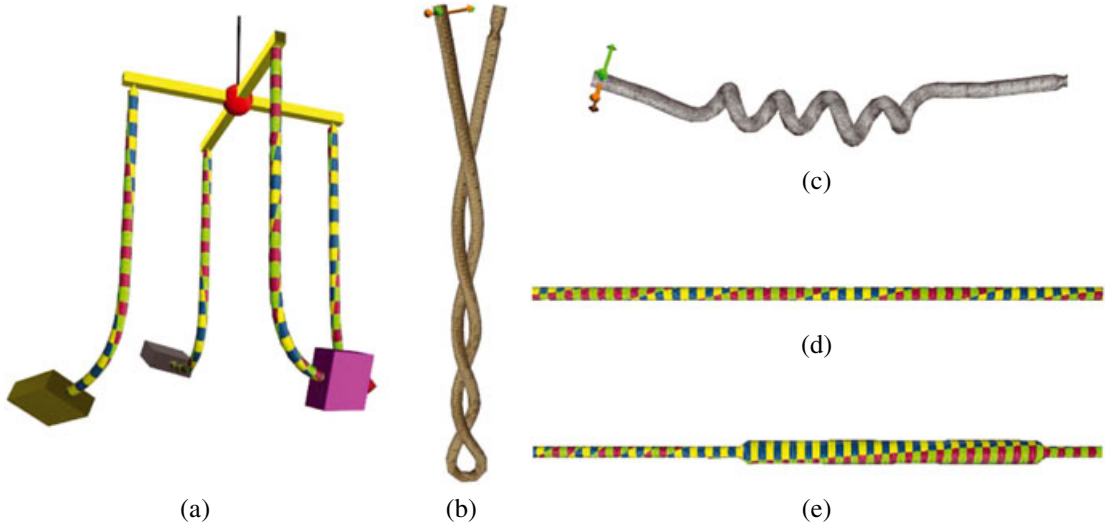


Figure 6. Our simulation results for the twisting effects. (a) An application for hanging boxes in wind forces. (b) The twisting effect of a string clamped at both ends. The string is gradually twisted on the left end and finally twisted to form a loop. (c) A twisted string that forms a spiral shape similar to a telephone cord. (d) A string with uniform torsional rigidity. (e) A string with non-uniform torsional rigidity.

L_i^{max} by moving $\mathbf{x}_i$ and/or $\mathbf{x}_{i+1}$ according to the state of a string. There are three possible cases.

1. A string clamped at one end: Length-constraint is applied from the clamped end sweeping through to another end. For example, in a string clamped at $i = 0$, $\mathbf{x}_{i+1}$ is moved, sweeping from the clamped end to another end.
2. A string clamped at both ends: We perform multiple adjustments from one end to another end of the string, sweeping back and forth repeatedly because correcting the length of one segment may change the length of other segments.
3. A string clamped at multiple points: We separate the string into strings clamped at both ends and strings clamped at one end and perform the length-constraint accordingly.

5.3. Tension Estimation

As previously stated, because of the constraint on lengths unrelated to applied forces, actual tensile stress and strain values cannot be directly computed from the simulation result. Here, we propose a novel approach to estimate the actual tensile stress and strain values for inextensible strings. The stress and strain are then used in handling elasticity, plasticity, tearing, and flicking of strings.

The actual tensile stress and strain values can be computed by estimating the *tensions* in the string. To derive the tensions, we also consider the particle positions computed *without* strain limiting. We model the *tension* $\mathbf{T}_i$ of segment i as a stiff force (Figure 4(d)) that makes its particles i and $i+1$ at both ends move from their unconstrained positions $\mathbf{x}'_i$ and $\mathbf{x}'_{i+1}$ (Figure 4(b)) to the constrained positions $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ (Figure 4(c)). In our implementation, we compute the tension as follows:

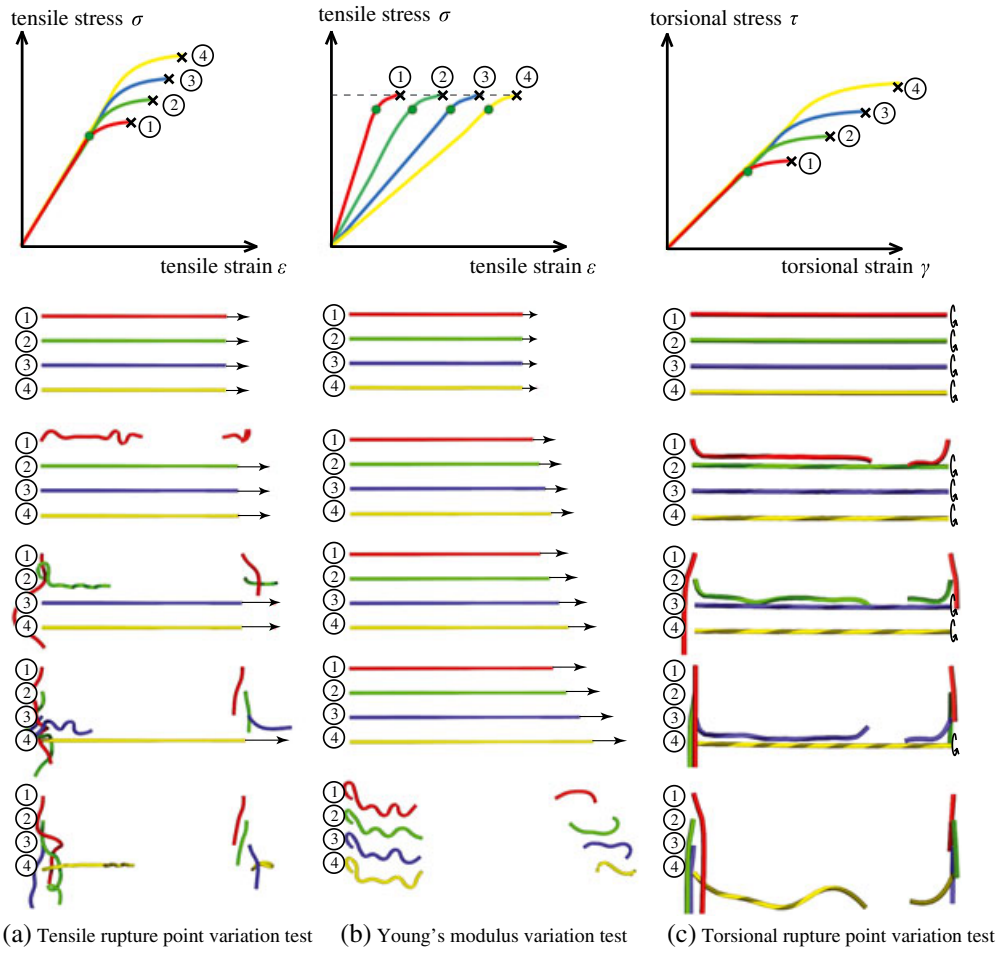


Figure 7. Experiments of various stress–strain curves and results of curves. A variety of stress–strain curves are shown at the top row, and their results are shown as animation sequences from top to bottom. (a) Tensile rupture point variation test; (b) Young's modulus variation test; and (c) torsional rupture point variation test.

DOI: 10.1002/cav

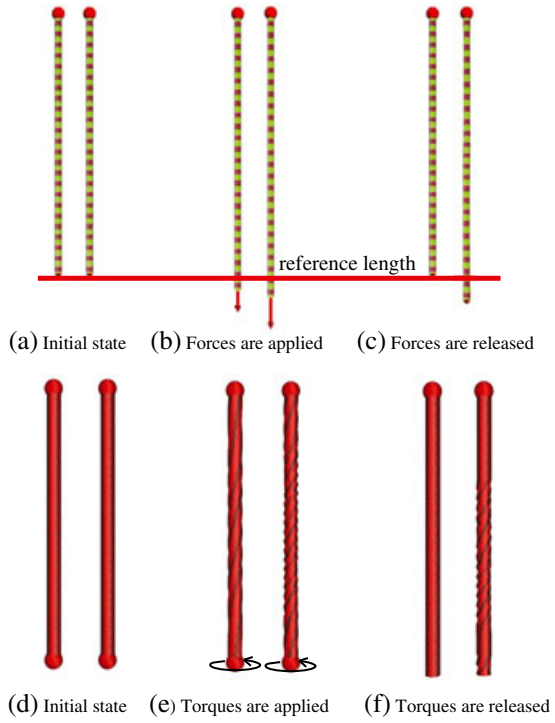


Figure 8. Plasticity of the material under tensile and torsional stresses. (a) and (d) are the states before forces and torques are applied in (b) and (d), respectively. In (b) and (d), the strings on the left do not pass the yield points, whereas the strings on the right do. (c) and (e) are the states after the forces and torques are released.

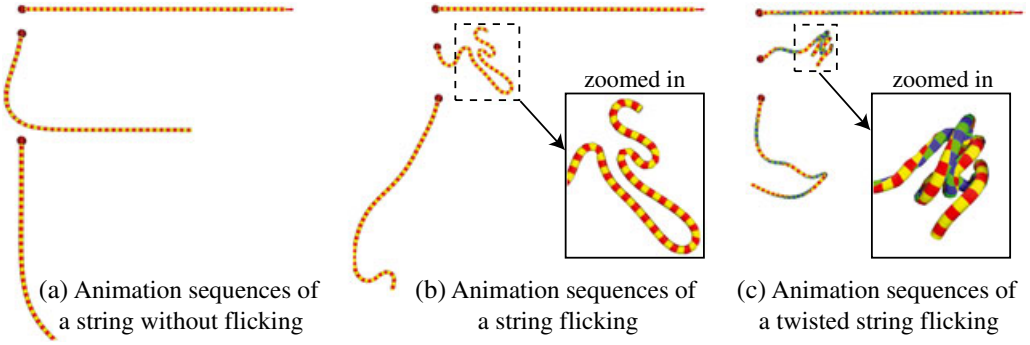


Figure 9. Flicking animation sequences of strings from top to bottom: (a) animation sequences of a string without flicking; (b) animation sequences of a string flicking; and (c) animation sequences of a twisted string flicking.

$$\mathbf{T}_i = k_{stiff}\left(\|\mathbf{x}'_{i+1} - \mathbf{x}'_i\| - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|\right)\mathbf{t}_i \qquad (12)$$

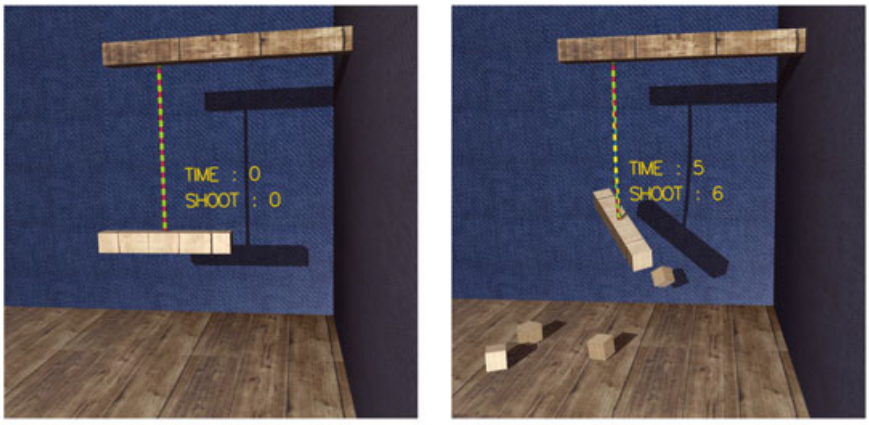
where k_{stiff} is a coefficient and $\mathbf{t}_i$ is a unit vector from particle i to $i+1$. The tension derived this way is used to reproduce tearing and flicking, as well as plastic behaviors of a string.

5.4. Tearing and Flicking a String

For tearing under a tensile stress, we assign a rupture point or a tensile stress threshold $\sigma_{rupture}$ for each segment. If the segment's tensile stress exceeds its tensile stress threshold, the segment will be broken. The applied tensile stress σ_i can be computed from tension $\mathbf{T}_i$ in each segment by using Equation (8) with $\mathbf{F}_n = \mathbf{T}_i$.

Similarly, we can handle the behavior of flicking by using the tension. When an inextensible string is pulled and released or torn apart, the applied stress vanishes, but the tensile strain of the segment from the elongated length still remains. The bouncing back force could be computed from an internal tensile stress translated from the tensile strain by referencing the stress–strain curve. However, with our tension estimation technique, we can directly use the tension as the bouncing back force. Note that, without this technique, the string would just fall down quietly by the

Figure 10. Animation sequences of a hanging bridge colliding with incoming crates.



(a) At the beginning of a game. (b) At the 5th second.

(c) At the 16th second. (d) At the 49th second. The string is torn by twisting.

Figure 11. This is a game application using our method. The goal of this game is to shoot a box and make a string twist until it is torn apart. The more it is twisted, the harder it will twist back. Therefore, if a player misses, the bar will decrease the twisting angles the player has hit so far. Players can compare the time and the number of boxes they used with others. (a) At the beginning of the game; (b) at the fifth second; (c) at the 16th second; and (d) at the 49th second, the string is torn by twisting.

DOI: 10.1002/cav

Table I. The computational time in milliseconds of each process in one time step. The time step in our implementation is 0.01 second.

Number of segments	Updating particles	CSM	Twisting computation	Tension estimation	Collision handling	Total time
100	0.011	0.086	0.098	0.221	1.31	1.73
150	0.022	0.168	0.184	0.424	1.36	2.16
200	0.029	0.221	0.237	0.564	1.37	2.42
746	0.128	0.501	0.739	1.67	3.13	6.17

CSM, chain shape matching.

force of gravity because tensile strain is limited and, thus, very small in an inextensible string.

As can be seen in the tensile stress–strain curve (Figure 3(a)), a real string is lengthened according to the tensile stress in the elasticity and plasticity states prior to the rupture point. Therefore, the maximum length L_i^{max} of each segment used in strain limiting should be updated accordingly (otherwise the string does not elongate). For this, we look up the tensile strain ε_i corresponding to applied stress σ_i from the tensile stress–strain curve and use it to compute the appropriate value of L_i^{max} by using Equation (10), $L_i^{max} = \varepsilon_i L_0 + L_0$. In the elasticity state, the tensile strain ε_i of a string becomes zero when applied forces are removed. In other words, the string returns to its original length. However, when its tensile strain exceeds the yield point (plasticity state), ε_i will remain the same as the last time the forces are applied. Our method also modifies the radius of a segment in order to preserve the volume of the segment when stretched.

Tearing of the string also happens when a sufficient torque is applied to the string. Tearing caused by twisting regularly arises in several soft strings such as spaghetti and licorice(candy stick). In contrast to tearing by stretching, torsional strain is not limited, and it can be directly calculated from current twisting angles (Equation (11)). Similar to tearing by stretching, when γ_i of segment i reaches a rupture point, we tear the segment. For a plasticity of twisting, when γ_i passes a yield point, we update θ_i^0 with $(\theta_{i-1} - \theta_{i+1})/2$ to make the twisted angle of segment i permanent.

6. COLLISION HANDLING

In this section, we introduce an optimized searching scheme for collision detection of strings, although previous works often use techniques based on the bounding volume hierarchy [2,17,18]. Apart from the bounding volume hierarchy, space partitioning using a grid-based data structure is a simple and efficient technique for collision detection of strings that have a large number of self-collisions. Specifically, we treat each segment as a capsule (a cylinder with two spheres at both ends) and search for capsule collision pairs. For neighbor searches, we use a uniform grid of voxels. The number of voxels to be searched is $27(= 3 \times 3 \times 3)$ in a naïve approach. For better performance, we found that it is sufficient to search for colliding segments in only seven neighboring voxels (top, bottom, left, right, front, back, and center voxels) under the following three specifications:

(1) specifying the voxel size equal to or larger than segment length l;
(2) storing indices of particles in each voxel; and
(3) searching for capsule collision pairs from two adjacent segments of each particle in the seven neighboring voxels.

For a better understanding, we describe it by using an example in a two-dimensional case (five neighboring cells). The idea can be generalized to the three-dimensional case in a straightforward manner. In Figure 5, Particles A and B are neighbors. Our method performs the segment collision test between their two adjacent segments, that is, pairs of segments $\{a, g\}$, $\{a, f\}$, $\{b, g\}$, and $\{b, f\}$. If two segments have an intersection, there is definitely a pair of both ends' particles residing in each of the other seven neighboring cells. This can be easily proved if one writes all possible cases in two dimensions, with five neighboring cells (center, up, down, left, and right).

The closest points of a pair of colliding segments i and j are indicated by fractions $s \in [0, 1]$ and $t \in [0, 1]$, respectively. In order to move the colliding segments to the non-intersection positions, we compute a displacement vector between the closest points. Then, we move both end particles of each segment, corresponding to the fractions s and t similar to [18].

Moving a colliding segment may make the string discontinuous, and thus, we repeat shape matching until particle positions converge. Conversely, shape matching may cause a collision again. As a result, iterations are required for both shape matching and collision constraints. To lessen the iterations, we temporarily make the masses of colliding particles heavier so that shape matching barely moves the particles. In our experiments, by making the colliding particles three times heavier, one iteration is sufficient.

7. RESULTS

Our implementation was written in C++ with OpenGL. All experiments were conducted on a desktop PC with an Intel Core i7 3.20GHz CPU and 6GB RAM. See

the accompanying video for the animation of our results (stringCAVW_lo.mov).

Figure 6 demonstrates the twisting effects in our model. An application for hanging boxes is presented in Figure 6(a), where objects at the tips of strings are rotated by wind forces, making the strings twisted. With twisting effects, the strings try to twist back to the initial state, making the rotational velocities increase and the objects roll back and forth in the wind. Twisting of strings can reproduce phenomena such as an instability of bending and twisting called *buckling*, which makes a string form a spiral shape (Figure 6(b) and (c)). Figure 6(d) and (e) shows the twisting of strings with uniform and non-uniform torsional rigidities, respectively. In the string with non-uniform torsional rigidity (thicker at the middle of the string in this result), the thicker part has a larger torsional rigidity and, therefore, has less twisting.

Our method enables the simulation of strings with various material properties, as shown in the variation tests in Figure 7. The stress–strain curves are shown in the top row, and their corresponding results are shown as animation sequences below each curve. Parameters used in tearing, that is rupture points, yield points, and Young'modulus, are assigned to all segments in the string. However, that kind of completely uniform strength is impossible in the real string, so we randomly altered the parameters in each segment with a range of variation up to 0.01%. Three experiments were conducted as follows:

(1) Tensile rupture point variation (Figure 7(a)): Tensile rupture points of strings are varying, increasing from string number 1 to 4. As expected, the topmost string (1), which has the lowest rupture point, is torn first.
(2) Young's modulus variation (Figure 7(b)): Young's modulus is a measure of the elasticity of a material. Because values of the Young's modulus of strings in this test are lessened from number 1 to 4, whereas the applied stresses required for breaking the strings are equal, the bottommost string (4) is lengthened the most before breaking.
(3) Torsional rupture point variation (Figure 7(c)): Similar to Figure 7(a), torsional rupture points are varying, increasing from string number 1 to 4. When the strings are twisted by the applied torques on the right-hand side, the topmost string (1), which has the lowest torsional rupture point, is torn first.

Figure 8 shows plasticity handling of strings in our model. When an applied stress from a force or torque passes a yield point, a string will irreversibly deform; this means the string will not return to its original length or twisting angle. In the top row, we compare two strings under the applied forces below (left string) and over (right string) the yield point. Likewise, two strings in the bottom row show a comparison under the different applied torques. It can be observed both in the top and bottom rows that the string on the left returns to its original shape, whereas the string on the right is permanently deformed.

Animation sequences of flicking are shown in Figure 9. Without flicking, the string in Figure 9(a) falls naturally when an applied force is removed. In our model, the string bounces back through the estimated tensions when the applied force is removed, as shown in Figure 9(b). When the twisted string in Figure 9(c) is pulled and released, the twisting effect also occurs.

For us to demonstrate the practical uses of our method, Figures 10 and 11 show applications in an animation and game. Figure 10 shows a destruction of a hanging bridge. Wooden boards (rigid bodies) are tied with strings (ropes in this case) to build the bridge. The ropes are gradually torn apart from collisions of the wooden boards and incoming crates that cause high tensions in the ropes. We used a particle-based simulation method [19] for rigid body simulation in our implementation. Figure 11 shows a twisting game. A box is hanging from a wooden beam by a string. The rule of the games is to shoot the box and make the string twist until it breaks.

The breakdown computational time in each process for strings with different numbers of particles is shown in Table I. All strings consist of 100 segments, except for the 150 segments in Figure 9, 200 segments in Figure 6(c), and 746 segments in Figure 10. The computational time of the results in Figures 10 and 11 is measured, excluding the time for the rigid body simulation.

8. CONCLUSION AND FUTURE WORK

We have introduced a simple model to simulating twisting, tearing, and flicking of strings, that is fast, easy to implement, and applicable to traditional simulation models. We have demonstrated that our method can handle twisting effects of strings with both uniform and non-uniform torsional rigidities, as well as rotational velocity caused by twisting.

By using our method, the tension in an inextensible string can be estimated for the generation of tearing and flicking effects. We have also enabled the tearing effects in twisting. A variation in the quality of strings, that is elasticity and plasticity, can be achieved. Although our method is not physically-based, it can successfully reproduce the interesting behaviors of strings which would greatly enrich the realism of interactive applications such as games.

Our method has some limitations. As previously mentioned, our method is not a full physically-based model. Thus, more advanced physics behaviors such as spring-twisting pendulum and anisotropic bending in [8] are hard to simulate. The rapid motion could cause the strings to pass through each other or themselves. However, this problem did not occur in our experiments. In case of rapid motion, continuous collision detection should be considered.

For future work, we would like to consider a deformation of cross-sections during twisting. In our model, each segment is rendered as a spherical cylinder. However, cross-sections of most materials can be deformed when it

is twisted. The non-uniform torsional rigidity in our model is considered along the length, not the cross-sections. A non-uniform density distribution within the cross-section should be considered to simulate more interesting results. The collision between segments is treated as a collision between rigid segments. We would like to improve the collision detection algorithm to handle the collisions between deformable segments. We also would like to improve the overall performance with a GPU implementation.

ACKNOWLEDGEMENT

This work was supported by Grant-in-Aid for JSPS Fellows (22·4748).

REFERENCES

1. Rosenblum RE, Carlson WE, Tripp III E. Simulating the structure and dynamics of human hair: modeling, rendering and animation. *The Journal of Visualization and Computer Animation* 1991; **2**(4): 141–148.
2. Selle A, Lentine M, Fedkiw R. A mass spring model for hair simulation. *ACM Transactions on Graphics* 2008; **27**(3): 64:1–64:11.
3. Hadap S. Oriented strands: dynamics of stiff multibody system, In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Vienna, Austria, Sept 2-4, 2006; 91–100.
4. Rivers A, James D. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics* 2007; **26**(3): 82:1–82:6.
5. Rungjiratananon W, Kanamori Y, Nishita T. Chain shape matching for simulating complex hairstyles. *Computer Graphics Forum* 2010; **29**(8): 2438–2446.
6. Bertails F, Audoly B, Cani M-P, Querleux B, Leroy F, Lévêque J-L. Super-helices for predicting the dynamics of natural hair. *ACM Transactions on Graphics* 2006; **25**(3): 1180–1187.
7. Spillmann J, Teschner M. Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects, In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, USA, Aug. 3-4, 2007; 63–72.
8. Bergou M, Wardetzky M, Robinson S, Audoly B, Grinspun E. Discrete Elastic Rods. *ACM Transactions on Graphics* 2008; **27**(3): 63:1–63:12.
9. Provot X. Deformation constraints in a mass-spring model to describe rigid cloth behavior, In *Graphics Interface 1995 Conference Proceedings*, Quebec, Quebec, Canada, May 17-19, 1995; 147–154.
10. Baraff D, Witkin A. Large steps in cloth simulation, In *ACM SIGGRAPH 1998 Conference Proceedings*, Orlando, USA, 1998; 43–54.
11. Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. *Journal of Visual Communication and Image Representation* 2007; **18**(2): 109–118.
12. Goldenthal R, Harmon D, Fattal R, Bercovier M, Grinspun E. Efficient simulation of inextensible cloth. *ACM Transactions on Graphics* 2007; **26**(3): 49–55.
13. Diziol R, Bender J, Bayer D. Volume conserving simulation of deformable bodies, In *Eurographics 2009 Short Papers*, Munich, Germany, March 30 - April 3, 2009; 37–40.
14. Rungjiratananon W, Kanamori Y, Metaaphanon N, Bando Y, Chen B-Y, Nishita T. Twisting, Tearing and Flicking Effects in String Animations, In *Proc. of Motion in Games 2011 (LNCS7060)*, 2011; 192–203.
15. Metaaphanon N, Bando Y, Chen B-Y, Nishita T. Simulation of tearing cloth with frayed edges. *Computer Graphics Forum* 2009; **28**(7): 1837–1844.
16. Bhuvenesh CG, Rajesh DA, David MH. *Textile Sizing*. CRC Press., New York, USA, 2004.
17. Sobottka G, Weber A. Efficient bounding volume hierarchies for hair simulation, In *Proceedings of the 2005 workshop on virtual reality interaction and physical simulation*, 2005; 1–10.
18. Spillmann J, Teschner M. An adaptive contact model for the robust simulation of knots. *Computer Graphics Forum* 2008; **27**(2): 497–506.
19. Harada T. Real-time rigid body simulation on GPUs. *GPU Gems 3, Chapter 29* 2007: 123–148.

AUTHORS' BIOGRAPHIES

Witawat Rungjiratananon received his BSc degree in computer engineering from Chiangmai University, Thailand, in 2006 and his MSc degree in computer science from the University of Tokyo, Japan in 2009. He is currently a PhD student in the Graduate School of Frontier Sciences, University of Tokyo and expects to graduate in September 2012. His research interests mainly focus on computer graphics and interactive simulation systems.

Yoshihiro Kanamori received his BSc, MSc, and PhD degrees in computer science from the University of Tokyo, Japan, in 2003, 2005, and 2009, respectively. Since April 2009, he has been an assistant professor in the University of Tsukuba. His research interests center on computer graphics including real-time rendering, point-based graphics, and interactive simulations.

Napaporn Metaaphanon recieved her BEng in computer engineering from Chulalongkorn University, Thailand, in 2005. From July 2005 to February 2008, she worked as a software developer at Massive Software. Then, in 2010, she received her MSc in computer science from the University of Tokyo, Japan. In October 2010, she started working as an R&D engineer at Square Enix. Her research interests are mainly related to computer graphics and physical simulation.

Yosuke Bando received his BSc, MSc, and PhD degrees in computer science from the University of Tokyo in 2001, 2003, and 2010, respectively. After joining TOSHIBA Corporation in 2003, he has been engaged in the development of advanced semiconductor chips for computer graphics, vision, and image processing.

Bing-Yu Chen received the BSc and MSc degrees in computer science and information engineering from the National Taiwan University, Taipei, in 1995 and 1997, respectively, and received his PhD in information science from the University of Tokyo, Japan, in 2003. Currently, he is working as a professor in the Department of Information Management, the Department of Computer Science and Information Engineering, and the Graduate Institute of Networking and Multimedia of the National Taiwan University, and also in the Department of Complexity Science and Engineering of the University of Tokyo. His research interests include mainly computer graphics, image and video processing, and human-computer interaction.

Tomoyuki Nishita is a professor in the Department of Complexity Science and Engineering (also in the Department of Information Science) at the University of Tokyo, Japan since 1998. He received his BEng, ME, and PhD in engineering in 1971, 1973, and 1985, respectively, from Hiroshima University. He taught at Fukuyama University from 1979 to 1998. He was an associate researcher in the Engineering Computer Graphics Laboratory at Brigham Young University from 1988 to 1989. His research interests center on computer graphics including lighting/shading models (radiosity), natural phenomena, real-time rendering, geometric modeling, and nonphotorealistic rendering.

DOI: 10.1002/cav