Physical Material Editing With Structure Embedding For Animated Solid

Ning Liu* Peking University
Xiaowei He† Institute of Software, Chinese Academy of Sciences
Yi Ren‡ Peking University
Sheng Li§ Peking University
Guoping Wang¶ Peking University

ABSTRACT

Physically-based soft bodies are difficult to animate for anisotropic and heterogeneous materials. Explicitly modeling a desired material behavior by tuning the constitutive parameters is a difficult, tedious and sometimes impractical task for an animator. Even with linear constitutive materials, there are still dozens of independent parameters to tune. In this paper, we propose a new technique to ease the animators' effort when modeling complex material behaviors. Our key idea is to treat the original complex material as the composite of a matrix and structure elements. These two constituent materials are both easy to simulate because the matrix is homogeneous and isotropic, while the structure elements have simple and intuitive deformation mode. In this way, complexity in tuning parameters is greatly reduced. By proper embedding structure elements into the matrix, an animator can design diverse and creative material behavior without tedious parameter tuning work. Our results illustrate that our approach is intuitive, easy-to-use and has potential for further extensions for physically-driven solid animation.

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

1 INTRODUCTION

During the last two decades, physically based simulation of solids has grown to be very popular in computer graphics and led to many application in games and visual effects industry. In contrast to the high accuracy demands in engineering, solid simulation in computer graphics calls more for easy-to-use interfaces, through which an animator can design complex dynamic material behaviors more conveniently.

However, designing physically-based animation by only considering the constitutive equations proves to be difficult due to the complex material properties. Simple material models such as St. Venant-Kirchhoff (StVK) [6] are easy to use (with only two independent parameters), but only applicable to homogeneous and isotropic solids. To model non-homogeneous and anisotropic materials, more accurate and complex constitutive models are needed. Unfortunately, these complicated models are not easy to use for an animator because of the large number of parameters. Even for linear anisotropic materials, there are still as many as 36 parameters in the stiffness/compliance matrix [14], tuning these not-so-intuitive parameters is non-trivial or even impractical work during an animation production.

*e-mail: liuning@graphics.pku.edu.cn
†e-mail:hexw@graphics.pku.edu.cn
‡e-mail:renyi@graphics.pku.edu.cn
§e-mail:lisheng@pku.edu.cn
¶Correspond to :wgp@pku.edu.cn

Graphics Interface Conference 2012
28-30 May, Toronto, Ontario, Canada

In this paper, we propose a novel method to overcome the above problem. Our work is inspired by the fact that many real world solids are composites which have a matrix as base material along with other inserted structure elements (see Figure 1). Based on this observation, we model a complex material as the combination of a matrix and several structure elements. The matrix is modeled as homogeneous and isotropic material which is relatively easy to simulate with StVK model, while the structure elements has simple but intuitive deformation mode which is easy to manipulate. After properly embedding structure elements into the matrix, the resultant material will show varieties of non-homogenous and anisotropic behaviors. To improve efficiency, we also develop a stroke based interface for better interaction.

Figure 1: Composite materials are made of two or more constituent materials with different mechanical behaviors and are used in engineering(left). The simplified model we use is made of a based matrix and embedded fibers.(right)

To the best of our knowledge, although geometric deformation tools have been widely used in computer graphics, no such compositing concept has been previously proposed to aid material design for physically-based animation. Our contribution can be summarized as:

- We propose a new concept of composite material to handle physically-based material design. We believe many complex behaviors of solid can be thus simulated by compositing relatively easy-to-model materials.
- We choose fibres as one type of structure element to illustrate the convenient usage of our method, and the result illustrated that it is intuitive, easy-to-use and barely pose any limit on the creativity of an animator, which will be of great help to an animator in designing physically-driven deformation.

Paper Organization: We will first give a quick literature review on related work in Section 2. After that, deformation background and our method will be detailed in Section 3 and Section 4. We introduce our user interface in Section 5. Results and implementation issues are addressed in section 6 after which a further discussion is presented in Section 7 before we draw a conclusion in Section 8.

2 MOTIVATION AND RELATED WORK

2.1 Material Modeling for Animation

Physically modeling the behavior of deformable material such as elasticity [9, 28], plasticity [25], fracture [26] is difficult and complex for computer graphics. To take a quick overview of the methods in graphics, please refer to two survey papers [24, 22]. Since the early work of [32], many approaches have been proposed to

model accurate behavior of a deformable object. What complicates the solid simulation is that real material has complex constitution. In most graphics applications, solid objects are only approximated with simple isotropic models such as St. Venant-Kirchhoff model [6]. However, a lot of solids deform highly anisotropically. These solids are usually consisted of multiple parts with distinctive material parameters, making them deform non-uniformly compared to simplified homogeneous materials. To model the complex material properties of a solid, a detailed material map is often used to describe material parameters for an arbitrary position inside the solid. In the work of [23], a high resolution finite element mesh with complex material information was simulated in the pre-computation step to calculate the coarse embedding. Similarly but based a meshfree framework, recent work of [10] used a detailed volumetric material map to compute the material-aware shape functions. With the help of the material map, the mechanical behaviors of a solid can be accurately simulated.

Usually, accurate material parameters are acquired from anatomy data for applications such as surgical simulation [23, 10] , or by measurement-based approaches [27, 5]. Yet for animation purpose, accuracy is not the most significant concern, the flexibility for designing desired material by a animator is of more importance. Unfortunately, designing the heterogeneous and anisotropic material is non-trivial work, due to the fact that the constitutive matrix for a linear anisotropic material contains at least 21 independent parameters([11]), which are hardly intuitive for animators to tune.

To solve the problem of material designing for animation, we use a novel method to avoid the difficulties in explicitly assigning the strain-stress relation parameters, thus ease the work for an animator. Our intuitive approach is based on a strategy that can be called structure embedding.

2.2 Embedding and Composites

The idea of embedding is widely used in computer graphics to enforce constraints during animation. It can be used to drive a high resolution mesh in a low resolution cage to avoid numerical problems and improve performance [9, 7]. It also helps circumvent the difficulty of handling mesh silvers during fracture [18]. [30] generalized the embedding techniques and demonstrated the ability to handle arbitrary refinement or resampling for collision processing. Later, [33] solved the limitation of non-negative weights in previous work using Procrustes transform. Embedding can also be implemented with constraint based methods [20]. The technique of linear skinning [15] can also be viewed as one kind of embedding. With rigid skeleton inside the model, the surface mesh deforms smoothly with pre-defined skinning weights.

In fact, many real world materials are composite materials made from two or more constituent materials with distinct physical properties. One kind of material made of a base material(which is called a matrix) and inserted fibers is widely used. It is known that solid of single material is weak for engineering use, but with short fibers embedded, the mechanics of the solid is greatly strengthened to be used as building materials[8]. One example is illustrated in Figure 1 showing a real world composite and a diagram illustrating the matrix-fiber composite model.

Our method is inspired by the composite materials mentioned above. Compared to previous approaches, our method provide more flexibility in modeling diverse material behaviors. Our method also distinguishes itself from previous work with heterogeneous and anisotropic modeling methods [13, 23], as we focus on developing an interactive and intuitive interface, aiming to aid the animators with complex physical materials design.

3 Basics

In this section, we will first give some quick review on elasticity modeling. When an elastic object moves from rest configuration

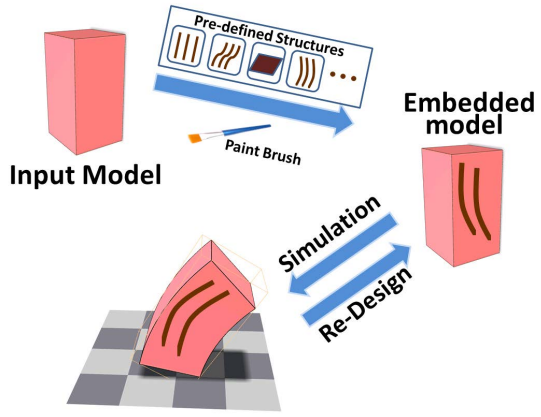

Figure 2: The work flow of our method. By the embedding technique with our interface(either pre-defined structures or painted by a user), physically-driven deformation can be designed by an animator easily and intuitively.

$\mathbf{X}$ to a new configuration $\mathbf{x}$, the deformation gradient $F = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}$ can be used to describe the deformation. This formulation is invariant under rigid translation but not rigid rotation. A better rotation invariant deformation measure can be derived as $\mathbf{E} = \frac{1}{2}(\mathbf{F^T F} - \mathbf{I})$ which is known as the Green-Lagrange strain tensor, with Cauchy strain tensor $E = \frac{1}{2}(\mathbf{F^T} + \mathbf{F} - \mathbf{I})$ as its linearized version. For linear materials, the relation between strain and stress tensor is assumed to be $\sigma = \mathbf{D}\varepsilon$ with $\mathbf{D}$ as a 6×6 constant coefficient matrix. By choosing different coefficients of the $\mathbf{D}$, heterogeneous and anisotropic materials can be simulated. If further isotropic assumption is made, $\mathbf{D}$ can be reduced to only two independent parameters: $\mathbf{D} = \mathbf{D}(\lambda, \mu)$ where λ and μ are the Lamé constants, or equivalent formula $\mathbf{D} = \mathbf{D}(E, \nu)$ where E and ν stand for Young's modulus and Poisson ratio. This kind of material is called St.Venant Kirchhoff (StVK) material and has grown extremely popular in computer graphics for its simplicity. The total elastic energy for the object can be integrated over the entire material domain Ω as

$$E = \int_\Omega \frac{1}{2} \varepsilon : \sigma d\Omega \tag{1}$$

where ":" stands for the sum of product of each pair of corresponding entries. To get the dynamics of the elastic deformation, equation

$$\rho \ddot{\mathbf{u}} + \mathbf{F_d}(\dot{\mathbf{u}}) - \frac{\partial E}{\partial \mathbf{u}} = \mathbf{F_{ext}} \tag{2}$$

is solved for each time step, where $\mathbf{u}$, $\mathbf{F_d}$, $\mathbf{F_{ext}}$ stand for displacement, damping force and external force respectively and $\frac{\partial E}{\partial \mathbf{u}}$ is the elastic force.

4 Methodology

In serious simulation such as medical simulation and surgical training, the coefficient matrix $\mathbf{D}$ can be loaded in from accurate tissue material data. Yet for animation purpose, $\mathbf{D}$ is designed by an animator seeking for desired deformation, and the designing process turns out to be complicated. Certain scenarios include: the animator would like the solid to be easy to bend than to stretch, or the solid has dramatic changes of material stiffness/compliance along some specific directions. For these kinds of material editing requirements, choosing proper $\mathbf{D}$ will be hardly intuitive or even impractical, because it's hard for an end user to associate the mechanical parameters with an experiential visual appearance.

We circumvent the difficulty of defining $\mathbf{D}$ by a technique called structure embedding. A structure is supposed to have simple but intuitive deformation modes. Low dimensional solids such as rod, plane can be taken as good examples. We start from an homogeneous and isotropic matrix material, then insert structure elements into the matrix which move with the matrix by geometric constraints. On the other hand, the structure elements also constrain the matrix material to move according to the structure deformation modes. In this way, we achieve an intuitive way to define how the matrix deforms.

To formulate our idea with more details, we present the total energy term as follows:

$$E = \int_{\Omega_M} \frac{1}{2}\varepsilon : \sigma d\Omega + \int_{\Omega_S} E_{structure} d\Omega, \tag{3}$$

$$\mathbf{u}_S = \Psi(\mathbf{u}_M) \tag{4}$$

where Ω_M and Ω_S stand for the volume taken by matrix material and structure elements, such that $\Omega_M + \Omega_S = \Omega$. $\mathbf{u}_S$ and $\mathbf{u}_M$ represent the displacement of structure elements and matrix. $E_{structure}$ is the energy term of the structures. Specifically, with line element (or called fibers), this term can be divided into bending energy, stretching energy, and twisting energy. Ψ is the embedding function to map positions from the matrix to the embedded structure elements. In our implementation, we update the structure elements $\mathbf{u}_S$ with the matrix $\mathbf{u}_M$ according to the geometric embedding function Ψ instead of updating them explicitly, which is most of the case as long as there is no topology changes of the matrix.

Based on the above discussion, the solid is now treated as composite material instead of single material object. As the structure elements take some volume fraction of the original matrix and contribute to the total energy, the resultant elastic force will drive the object to behave in the way that the embedded structure elements deform.

To detail the technique, we take line structure as an example and discretize the above equations in a meshfree framework in the next section.

4.1 Modeling A Single Fiber

The basic structure element in our method is called a fiber. A fiber is a line shaped structure element which is modeled as short elastic rod. We make an assumption that a fiber is inextensible, which will be used in our later computation. Choosing fiber as the basic structure element has several advantages. First, the deformation mode(bend,stretch,twist) is relatively easy for human perception. Second, its simplicity makes it perfect for user interaction through a stroke editing interface and is easy to insert into the matrix than other shaped elements.

Several previous work has been presented on rod elasticity. The early work dates back to [32]. Recent years, methods are proposed based on implicit centerline representation such as hair modeling [4]. [31] uses quaternionic material frame formulation and [3] is built on natural Bishop frame, these methods share the similarity of explicit centerline representation. More recently, geometrical matching [29] methods and unified treatment [16]are also applied in rod simulation.

We model a fiber as short elastic rod using the explicit centerline similar to [3], but with a simplified version which only considers bending and stretching energy, due to the fact that they are the most notable energy terms in our embedding framework.

To discretize a single fiber, the centerline is represented as connected line segments with $n+2$ vertices $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{n+1}$ and $n+1$ edges $\mathbf{e}^1, \mathbf{e}^2, ..., \mathbf{e}^n$ (figure 3(a)). The discrete curvature κ_i is measured by turning angle ϕ_i between two consecutive edges as below

$$\kappa_i = 2\tan\frac{\phi_i}{2}, \tag{5}$$

and the discrete curvature binormal vector is defined as

$$\kappa\mathbf{b}_i = \frac{2\mathbf{e}^{i-1} \times \mathbf{e}^i}{|\mathbf{e}^{i-1}||\mathbf{e}^i| + \mathbf{e}^{i-1} \cdot \mathbf{e}^i} \tag{6}$$

with the relationship $2\tan\frac{\phi_i}{2} = |\kappa\mathbf{b}_i|$ ([3]).

The bending energy of a single fiber can be integrated as

$$E_{bend} = \sum_1^n \frac{1}{2} K_b \frac{\kappa\mathbf{b}_i \cdot \kappa\mathbf{b}_i}{|\mathbf{e}^{i-1}| + |\mathbf{e}^i|}, \tag{7}$$

where K_b is the bending stiffness of the fiber. However, this formula only applies to fibers with straight configuration. To deal with the curved fibers, the turning angle $\frac{\phi_i}{2}$ in equation (5) should be replaced by $(\frac{\phi_i}{2} - \frac{\bar{\phi}_i}{2})$ where $\bar{\phi}_i$ presents the rest turning angle. For conciseness, we show the energy term for curved fiber in appendix.

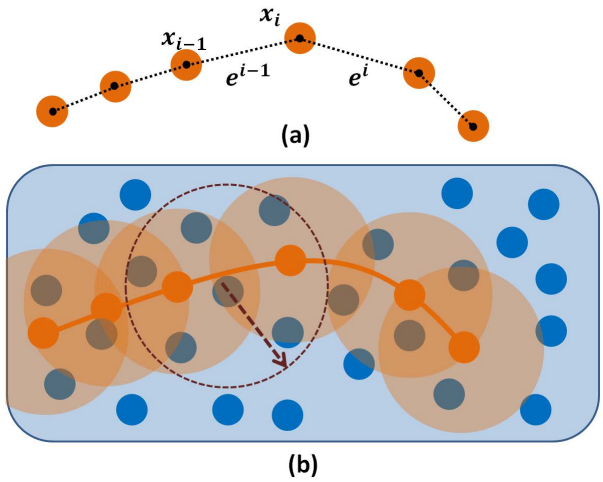

Figure 3: We discretize a fiber by their connected structure particles(SP) which form the fiber's centerline shown as orange dots(a). The matrix is discretized as cloud of material particles(MP) shown as blue dots (b). Both particles affect nearby particles through a kernel function decaying with distance.

To compute the dynamics of each discrete fiber, the gradient of bending energy $\frac{\partial E_{bend}}{\partial \mathbf{x}}$ must be computed. [3] computed the gradient through a technique called Automatic Differentiation in [12], we find that by taking the assumption of an inextensible fiber, direct approximated differentiation is easy to compute. We show the detail of the gradient in appendix for conciseness as well. The bending force for each discrete vertex can be derived once $\frac{\partial E_{bend}}{\partial \mathbf{x}}$ is computed. However, since the force is not symmetric between neighboring vertices, ghost forces will appear due to the violation of momentum conservation. We correct the result to get a symmetric form by adding react forces to two neighboring vertices shown in Figure 4.

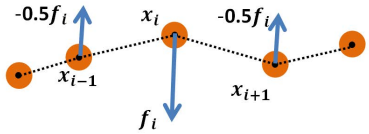

Figure 4: We add symmetry force to neighboring nodes to ensure momentum conservation at each fiber node(SP).

We handle the stretch of each fiber by enforcing a distance constraint with K_s as the stretch stiffness and $\mathbf{e}_0^i$ as the initial shape of $\mathbf{e}^i$ when animation begins.

$$E_{stretch} = \sum_1^n K_s(|\mathbf{e}^i| - |\mathbf{e}_0^i|)^2, \tag{8}$$

By taking negative gradient of energy, the bending/stretching force can be calculated to integrate the fiber particles, or their embedding matrix particles.

4.2 Modeling Fiber Embedding

The matrix is modeled as homogeneous and isotropic elastic solid in our framework. We choose to use a meshfree corotational form [2] for its simplicity and robustness with degenerated configurations. Compared with alternative methods such as Moving Least Square(MLS) method [21] which is limited by matrix inversion under degenerated conditions, corotational form handles degenerated configurations well. Besides, using corotational form enables the application of linear strain tensor without artifacts. For each particle i with its strain ε_i and stress σ_i, we can calculate its strain energy U_i as

$$E^i_{elastic} = V_i \frac{1}{2}(\varepsilon_i : \sigma_i) \tag{9}$$

where $V_i = m_i/\rho_i$ represent the volume of particle i. The elastic force that particle i exerts on its neighboring particle j can then be defined as the negative of the strain energy with respect to displacement.

$$\mathbf{F}_{ji} = -\nabla_{\mathbf{u}_j} E^i_{elastic} = -V_i(\mathbf{I} + \nabla \mathbf{u}_i{}^T)\sigma_i \mathbf{d}_{ij} \tag{10}$$

and $\mathbf{d}_{ij}$ is defined by

$$\mathbf{d}_{ij} = V_j \nabla W(\mathbf{x}_{ij}) \tag{11}$$

$$\nabla \mathbf{u}_i = \sum_j V_j \mathbf{u}_{ji} \nabla W(\mathbf{x}_{ij}) \tag{12}$$

where $\mathbf{u}_{ji}$ is the vector between neighboring particles i and j and weight $W(\mathbf{r})$ is defined as a smoothing kernel function

$$W(\mathbf{r}) = \begin{cases} 1 - 6(\frac{|\mathbf{r}|}{r_i})^2 + 8(\frac{|\mathbf{r}|}{r_i})^3 - 3(\frac{|\mathbf{r}|}{r_i})^4 & 0 \le |\mathbf{r}| \le r_i \\ 0 & |\mathbf{r}| > r_i \end{cases} \tag{13}$$

with r_i representing the support radius of node i.

$$\mathbf{u}_{ji} = \mathbf{u}_j - \mathbf{u}_i = \mathbf{R}^{-1}(\mathbf{x}_j - \mathbf{x}_i) - (\mathbf{x}_j^0 - \mathbf{x}_i^0) \tag{14}$$

$\mathbf{x}_i^0$ and $\mathbf{x}_j^0$ stand for the position vector in rest configuration. The rotation matrix $\mathbf{R}$ indicates the rigid rotation between the particle's current configuration and the rest configuration. $\mathbf{R}$ is computed through Polar decomposition of $\mathbf{A}_{pq_i}$ where

$$\mathbf{A}_{pq_i} = \sum_j V_j W(\mathbf{x}_{ij}^0)(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j^0 - \mathbf{x}_i^0)^T \tag{15}$$

Note that when implementing the elastic force above, we add react force to a particle's neighbor to ensure momentum conservation. To take fibers' energy into the total energy, we update the total energy after fibers are inserted into the matrix. As a result, the discrete nodes are divided into Material Particle(MP) which represent the matrix, and Structure Particles(SP) which build each structure element. The embedding function connecting each SP with nearby neighboring MP can be computed as a general shape function as

$$\Psi(\mathbf{x}) = \sum_i \phi_i(\mathbf{x})\mathbf{x}_i \tag{16}$$

where ϕ_i is the shape function of particle i for position $\mathbf{x}$ which can be computed with SPH or MLS approximation [21]. Here we choose the normalized weighting $\phi(\mathbf{x})$

$$\phi_i(\mathbf{x}) = \frac{W(\mathbf{x} - \mathbf{x}_i)}{\sum_j W(\mathbf{x} - \mathbf{x}_j)}. \tag{17}$$

Finally, the total energy for a particle i can be simplified as

$$E^i = (1 - \lambda_i)V_i E^i_{elastic} + \lambda_i V_i (E^i_{bend} + E^i_{stretch}) \tag{18}$$

In this equation, E^i represents the total energy of particle i's volume V_i. $E^i_{bend} + E^i_{stretch}$ evaluate the energy contributed by near-by structure elements. The volume fraction λ_i which measures the contribution of the structures can be estimated by the nearest structure particle to i. During animation, each MP is integrated through equation (2) using the total energy, and each SP is updated by the shape function $\Psi(\mathbf{x})$. The bending and stretching forces are computed at each discrete fiber positions, then we diffuse them to each of the neighboring matrix particles by $\Psi(\mathbf{x})$, momentum conservation is satisfied for the system according to symmetric calculation in Section 4.1. As the structure elements take some volume fraction of the matrix, the matrix will be driven to deform in order to conform to the structure's deformation mode, which demonstrate anisotropic characteristics. Under an extreme condition, the volume fraction of a fiber can be 1.0, indicating that the material deformation behavior will be totally determined by the embedded fibers. The volume fraction is reasonable because when we use fibers to reinforce the matrix, the resultant material can be either stronger or weaker, depending on the stiffness of the fiber. For material particles that have no neighboring structure particles, λ is simply set to be zero. In our implementation, the value of λ is a flexible user choice which makes the particle behave more "matrix" or "fiber". The animation process is shown in table 4.2 for summary.

	Algorithm Animation
1	**while** animating **do**
2	**for each** MP **do**
3	calc rotation from neighboring MP
4	calc elastic force from neighboring MP
5	**for each** fiber i
6	**for each** SP of fiber i
7	calc bending force from neighboring SP
8	diffuse bending force to neighboring MP
9	calc stretching force from neighboring SP
10	diffuse stretching force to neighboring MP
11	add collision force and external force for each MP
12	time integrate each MP
13	update SP by their embedded MP

5 User Interface

Stroke based interface is widely used among graphics community for image processing, mesh editing and is popular in commercial softwares such as MAYA and ZBrush. The line-shaped characteristics of fiber elements make them perfect for stroke interface. We use an interface to help the user place fibers inside a model.

After the uniform model(scanned or manually modeled) is loaded, three viewports are created with two of them showing orthogonal side view for drawing, and a third view for result checking and adjustment. Each fiber is generated by re-sampling two separate strokes in the viewport to get a three dimensional vertex sequence. This interface does not solve a 2D-3D mapping perfectly, but works for most cases when considering short fibers. To give further adjustment, we also develop controls to translate/rotate a fiber and each of its vertices manually as well. Note that the stroke does not have

to be precisely inside the triangle surface, because the simulation nodes are generally sparser than the triangle surface for computational efficiency. Our user interface computes the embedding weights and discards far-away strokes automatically.

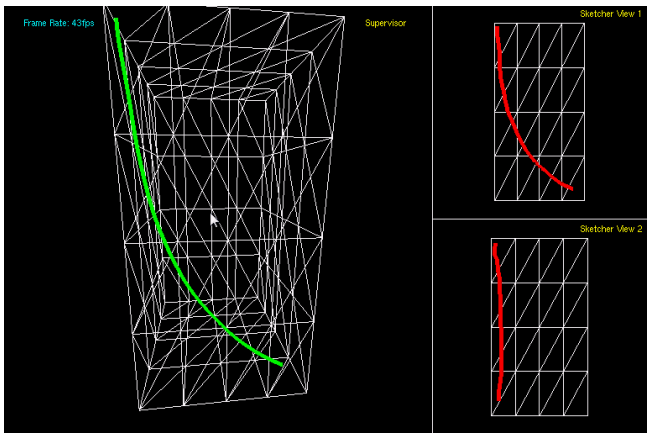

Figure 5: A user takes one stroke on each of two orthogonal views on the right, the interface generates the fiber shape in the left view. Further translation and rotation can be made for adjustment by the user to place the fiber at a desired position. This figure shows the process of fiber embedding for generating twist deformation in Figure 6.

6 Implementation and Results

We implement our method upon a meshlfree framework for their simplicity and intuitive demonstration. Traditional techniques such as spatial hashing and GPU accelerated skinning [1] can be easily implemented. Generally the average neighbors of each particle are adjusted to be 30 to 60. For time integration, choice can be made between implicit and explicit methods. Although an implicit solver can be used to support unconditionally stability and larger time step, we find explicit Leap-Frog method sufficient to give interactive feedback during animation. Compared to implicit method, an explicit one is very fast, thus can be looped multiple times between subsequent rendering frames to guarantee interactive frame-rate. Meanwhile, the stability problem can be alleviated using damping forces. In our implementation, time step is from 1×10^{-5} to 5×10^{-4} and simulate 50 to 100 times between rendering frames will still allow interactive speed.

For rendering, a common technique is to embed a detailed surface mesh into the simulation mesh, and move the vertices with the simulated nodes. Generally barycentric weights are used in tetrahedra meshes [19] and MLS shape function are applied in meshfree methods [1, 16]. We compute the same weights as equation 17 for each vertices when initialization and update the positions with them during simulation.

Figure 6 shows a short bar which deforms after it drops to the floor, the transparent version displays the embedded fiber layout. Normally, an homogeneous and isotropic bar will bounce up and down in a boring way. But with structure elements embedded, the bar shows distinct deformation mode: The fibers can be used to strengthen the material along one direction (left), or used to force the solid to shear/twist deformation(middle/right). Further contrast is shown in Figure 7. A long bar is fixed on the floor and pushed down by a heavy panel. Our embedding technique allows an intuitive way to guide the solid towards the user's desired deformation mode.

Figure 8 and 9 showcase the application of the our method in a plant model swinging caused by an initial impulse. The plant model has many different materials: the stem is stiffer at the bottom but more compliant at the top; the leaf is less stiff than the stem. Modeling the plant with isotropic material will fail for two reasons. First, a over-compliant plant will not reflect the inside structures and easy to fall down, while a over-stiff plant will appear unrealistic rigid without deformation details. Second, isotropic material generate incorrect deformation along of stems and veins, which results in unrealistic stretching appearance(please check our accompanied video for detail). Our method overcomes this problem by allowing a uniform built model as input. Afterward, by simply several strokes, the user can strengthen the plant by inserting fibers into the stem and leaf vein. To distinguish between bottom and top part of the stem, the user can further embed more fibers at the bottom. As a result, the plant moves and deforms in a reasonable way without incorrect stretching deformations along the stem. In general cases, when an animator wants to strengthen part of the solid, he can achieve the result by embedding fibers in that local area very easily withs our method.

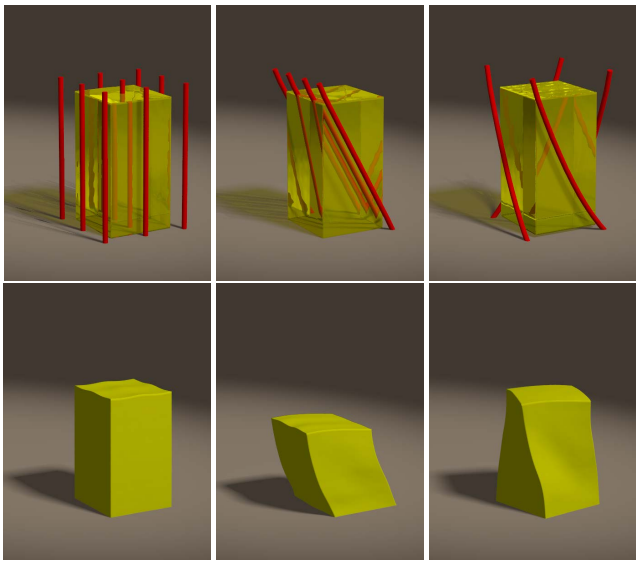

Figure 6: A short bar drops and collides with the floor. With different fiber layout, the bar illustrates distinct deformation behaviors. The figure shows vertical strengthen, shear deformation and twist deformation respectively.

As a meshfree framework, the kernel function is important to the stability and efficiency. We choose the kernel range ($h = 0.015m$) and initial particle space ($0.4h$) to guarantee enough number of neighbors (30 to 60) exist for each particle. Each particle has a fixed mass of $4 \times 10^{-4} kg$. Further parameters for the test scene in the paper are shown in Table 1. The timings in our experiment show that the computation time for structure elements only take less than 5% of the simulation time(not including collison and surface updating). This indicates that the fiber embedding can be integrated into a existing framework with only minimal additional cost. Our un-optimized code is currently run on a PC with 3.0Ghz CPU and 6G RAM, parallelling the code with MPI or graphics hardware is a future work.

7 Further Discussion

Why it works & How to use it

The motivation of our method is based on two facts: The isotropic and homogeneous material is easy to simulate, but generates boring uniform deformations. The structure elements such as fibers have simple but intuitive anisotropic deformation mode, which

Table 1: Scene complexity and timing.

Model	#Vertex/Face	#MP	#Fiber/SP	Young's./Poisson	Kb	Ks	Elastic Calculation	Bend&Stretch calculation
Short Bar	24.6k/49.1k	588	4/52	5e7/0.33	5e5	2.5e5	1.755ms	0.072ms
Long Bar	5.8k/11.6k	375	2/28	3e8/0.33	1e4	2.5e5	1.067ms	0.045ms
Plant	31.4k/47.2k	2723	5/90	1e5/0.33	1e5	1.8e5	18.750ms	0.235ms
Leaf	3.9k/7.4k	1159	5/86	2e8/0.33	3.5e6	3e5	7.422ms	0.198ms

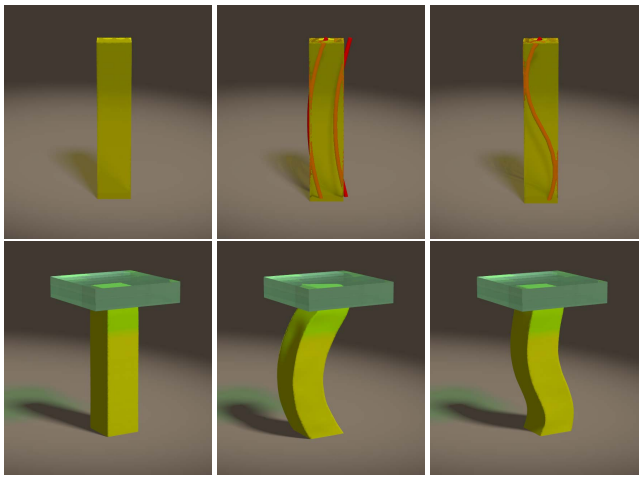

Figure 7: A long bar is pushed down with its end fixed on the floor. By embedding different shaped fibers, distinct bending deformations are generated in a physically-driven way.

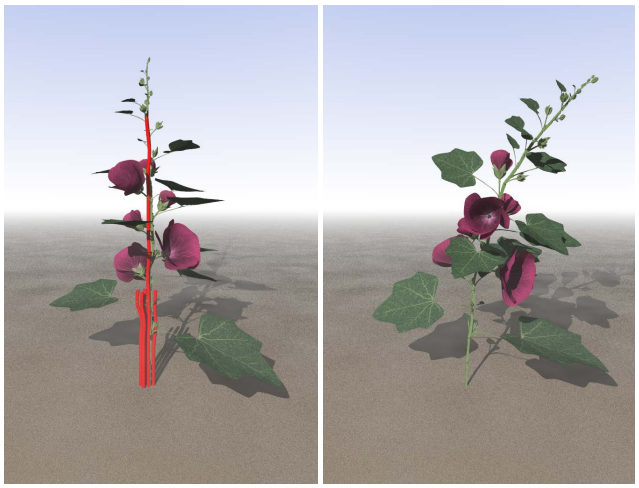

Figure 8: A plant swings caused by an initial impulse. As isotropic model fails and generates unrealistic deformation along the stem, we add several fibers to distinguish different material parts and achieve reasonable result. (please check our accompanied video for detail.)

make them easy to direct. Combining these two together, we can simulate non-homogeneous and anisotropic behaviors quite easily. This idea takes advantages of the relatively easy-to-simulate models to generate complex models, thus avoids the tedious process to build a complex model directly. Note that homogenization [13] might be possible for the resultant material, but we argue that for animation purpose, accurately calculating the parameters for a composite is not the primary interest for an animator, and using simple structures embedding is sufficient for most cases. As a structure element, a fiber is simple to manipulate for its geometry, and also deforms with strong directional bias (allow bending but restrict stretching). According to the energy of composite equation 18, a fiber can either strengthen or weaken the material by the volume fraction. The volume fraction can be computed based on the distance between a MP and a SP, or can be assigned directly by the user. Although a fiber is represented as line-shaped element which is easy to sketch, we can use multiple fibers to generate volumetric effect and build pre-defined patterns to give the user a visual guide on possible deformations(Figure 2), with these examples, the user can learn and design his own fiber layout for desired deformation. After all, our framework hardly sets any limit on the fiber layout and allows creative work designed by an animator.

Comparison with other tools

Traditional tool sets such as deformation tool in MAYA can also define bend/squash/twist deformation, but these methods are basically geometric methods which take no physical material property into consideration. Recent work of [17] also showed interesting results guiding the deformation to the user defined example shapes. In contrast to their method which focuses on interpolating between the example shapes at discrete frames, our approach tries to build the model from the simple and intuitive structures that directly drive the deformation. Furthermore, their method needs a set of input example shapes which are built using additional mesh editing tools, while our method is quite intuitive and stand alone. Our method also shares similarities with constraint based dynamic systems such as [7], because the fiber actually constrains the stretching deformation along its direction. But as the structures are intrinsically deformable solid, their deformation behavior can be well perceived by an animator which we believe is more intuitive to use than the mere geometric constraints.

Extensibility

Although we only consider bend and stretch energy for each fiber for their significance, other energy term such as twist energy can be implemented as well to enable more special effects, we leave them for the future work. The structure elements are also not limited to just fibers. As each structure element is discretized as structure particles(SP), other structures e.g. plane or shell or any other user defined structures can be implemented in the same way as a single fiber, enabling the simulation of diverse soft bodies. By defining void structures which have zero energy but cut in the way of the matrix, porous material might be simulated as well. The meshfree framework is used to demonstrate our method's effectiveness, but the composite idea is applicable to finite element methods as well, with the changed form of the shape function for embedding. Moreover, we have only modeled the elastic behavior of animated solid by manipulating the structure elements, but it will be an appealing work to consider other physical behaviors such as plasticity, viscosity and fracture as well. As an example in Figure 10, we illustrate the napping effect of a leafstalk by simply subdividing the embedded structure into two sub-structures.

8 Conclusion

We have introduced the composite material concept to model complex deformation of animated solid. The key idea is to model the

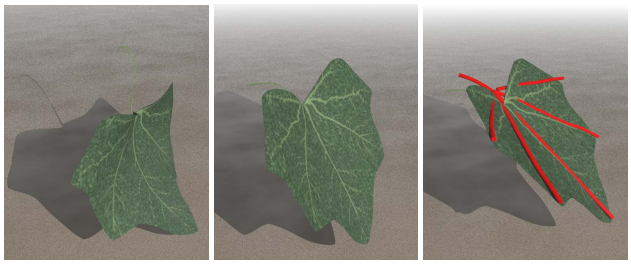

Figure 9: A leaf sags under gravity. The embedded fibers easily capture the structure information of the vein of the leaf and result in reasonable result. A over-compliant leaf fails to sustain the strength (left). A over-stiff leaf moves rigidly causing deformation detail loss (middle). Our method strengthens the areas where fiber passes while preserve deformation detail elsewhere (right).

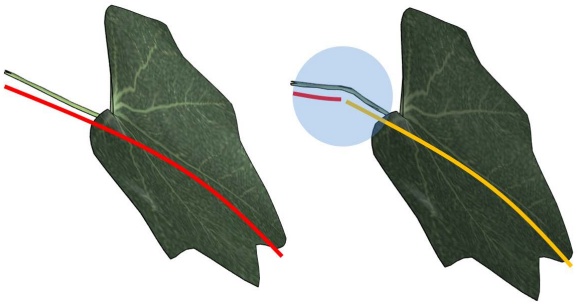

Figure 10: A structure element can change topology. When the embedded fiber break into two sub-fibers, bending energy become discontinuous at the breaking point, which make the leaf stalk appear being snapped.

heterogeneous and anisotropic material by embedding simple structures into an isotropic matrix. We use fibers as example structure elements and demonstrate its convenient and intuitive uses. Our approach avoids the tedious parameter tuning process of directly modeling a complex material and allows an animator to design desired deformation in an intuitive and creative way. We believe this technique will greatly help an animator to design complex material for physically-based animation. For future work, more interesting structure elements can be explored to to model diverse and appealing material behaviors such as plasticity and fracture.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. We also would like to thank Liming Xu for the valuable discussions. This work was supported in part by a grant from No. 2010CB328002 the National Basic Research Program of China, and by No. 61170205, 60833007, 60925007, 61121002 from National Natural Science Foundation of China.

A Appendix: Gradient of the Bending Energy

According to the definition of curvature binormal vector $\kappa\mathbf{b}_i$ and the assumption of inextensibility, its derivative with component x_j for an fiber i is

$$\frac{\partial(\kappa\mathbf{b}_i)}{\partial x_j} = \frac{(2\xi_j\times\mathbf{e}^i)(|\mathbf{e}^{i-1}||\mathbf{e}^i|+\mathbf{e}^{i-1}\cdot\mathbf{e}^i)-2\mathbf{e}^{i-1}\times\mathbf{e}^i(\xi_j\cdot\mathbf{e}^i-\mathbf{e}^{i-1}\cdot\xi_j)}{(|\mathbf{e}^{i-1}||\mathbf{e}^i|+\mathbf{e}^{i-1}\cdot\mathbf{e}^i)^2},$$

in which ξ_j is the unit vector with j-th component valued 1.0. The gradient of bending energy can be computed as

$$\begin{aligned}\frac{\partial E^i_{bend}}{\partial x_j} &= K_b\frac{\kappa\mathbf{b_i}}{|\mathbf{e}^{i-1}|+|\mathbf{e}^i|}\cdot\frac{\partial(\kappa\mathbf{b}_i)}{\partial x_j}\\ &= \frac{4K_b}{L_i a_i^3}[(\mathbf{e}^{i-1}\times\mathbf{e}^i)\cdot(\xi_j\times(\mathbf{e}^i+\mathbf{e}^{i-1}))a_i\\ &-(\mathbf{e}^{i-1}\times\mathbf{e}^i)\cdot(\mathbf{e}^{i-1}\times\mathbf{e}^i)(\xi_j\cdot(\mathbf{e}^i-\mathbf{e}^{i-1})))]\end{aligned}$$

with

$$\begin{aligned}l_i &= |\mathbf{e}^i|\\ L_i &= |\mathbf{e}^{i-1}|+|\mathbf{e}^i|\\ a_i &= |\mathbf{e}^{i-1}||\mathbf{e}^i|+\mathbf{e}^{i-1}\cdot\mathbf{e}^i\end{aligned}$$

The above equations only apply to fibers with straight initial shape. For a fiber with curved rest configuration, the discrete curvature will change to

$$\kappa_i = \tan(\frac{\phi_i}{2}-\frac{\bar{\phi}_i}{2})$$

with $\frac{\bar{\phi}_i}{2}$ the turning angle in rest state. For conciseness, we will use ϕ_i instead of $\frac{\phi_i}{2}$ to represent the turning angle in following calculation.

$$\kappa_i = \tan(\phi_i-\bar{\phi}_i)$$

According to the fact that

$$\tan(\phi-\phi_0) = \frac{\tan\phi-\tan\phi_0}{1+\tan\phi\tan\phi_0},$$

The bending energy is determined as

$$\widetilde{E^i_{bend}} = \sum_1^n\frac{1}{2}K_b\frac{4\tan^2(\phi_i-\bar{\phi}_i)}{|\mathbf{e}^{i-1}|+|\mathbf{e}^i|}.$$

Taking derivative with respect to x_j for each vertices leads to

$$\begin{aligned}\frac{\partial\widetilde{E^i_{bend}}}{\partial x_j} &= K_b\frac{\tan(\phi_i-\bar{\phi}_i)}{|e^{i-1}|+|e^i|}\frac{\partial\tan(\phi_i-\bar{\phi}_i)}{\partial\tan\phi_i}\frac{\partial\tan\phi_i}{\partial x_j}\\ &= K_b\frac{\tan(\phi_i-\bar{\phi}_i)}{|e^{i-1}|+|e^i|}\frac{(1+\tan^2\bar{\phi}_i)}{(1+\tan\bar{\phi}_i\tan\phi_i)^2}\frac{\partial\tan\phi_i}{\partial x_j}\end{aligned}$$

With the relationship of

$$4\tan^2\phi_i = \kappa\mathbf{b}_i\cdot\kappa\mathbf{b}_i$$

$$\tan\phi_i\frac{\partial\tan\phi_i}{\partial x_j} = \frac{1}{4}\kappa\mathbf{b}_i\cdot\frac{\partial\kappa\mathbf{b}_i}{\partial x_j}$$

the energy derivative of a curved fiber vertex can be computed as

$$\frac{\partial\widetilde{E^i_{bend}}}{\partial x_j} = \frac{\tan(\phi_i-\bar{\phi}_i)}{\tan\phi_i}\frac{(1+\tan^2\bar{\phi}_i)}{(1+\tan\bar{\phi}_i\tan\phi_i)^2}\frac{\partial E^i_{bend}}{\partial x_j}$$

where the energy term for straight rest $\frac{\partial E^i_{bend}}{\partial x_j}$ configuration has been computed in above equations.

REFERENCES

[1] B. Adams, M. Ovsjanikov, M. Wand, H. Seidel, and L. Guibas. Meshless modeling of deformable shapes and their motion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 77–86. Eurographics Association, 2008.

[2] M. Becker, M. Ihmsen, and M. Teschner. Corotated sph for deformable solids. In *Proc. Eurographics Workshop on Natural Phenomena*, pages 27–34, 2009.

[3] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete Elastic Rods. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):63:1–63:12, aug 2008.

[4] F. Bertails, B. Audoly, M. Cani, B. Querleux, F. Leroy, and J. Lévêque. Super-helices for predicting the dynamics of natural hair. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1180–1187. ACM, 2006.

[5] B. Bickel, M. Bächer, M. Otaduy, H. Lee, H. Pfister, M. Gross, and W. Matusik. Design and fabrication of materials with desired deformation behavior. In *ACM Transactions on Graphics (TOG)*, volume 29, page 63. ACM, 2010.

[6] J. Bonet and R. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge Univ Pr, 1997.

[7] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovic. Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics*, 21(3):586–593, 2002.

[8] T. Courtney. *Mechanical behavior of materials*. McGraw-Hill, 2000.

[9] G. Debunne, M. Desbrun, M. Cani, and A. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 31–36. ACM, 2001.

[10] F. Faure, B. Gilles, G. Bousquet, and D. Pai. Sparse meshless models of complex deformable solids. In *ACM Transactions on Graphics (TOG)*, volume 30, page 73. ACM, 2011.

[11] R. Feynman, R. Leighton, and M. Sands. *The Feynman lectures on physics*, volume 2. Addison-Wesley Reading, MA, 1964.

[12] E. Grinspun, A. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67, 2003.

[13] L. Kharevych, P. Mullen, H. Owhadi, and M. Desbrun. Numerical coarsening of inhomogeneous elastic materials. *ACM Transactions on Graphics (TOG)*, 28(3):51, 2009.

[14] W. Lai, D. Rubin, and E. Krempl. *Introduction to continuum mechanics*. Butterworth-Heinemann, 2009.

[15] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings of Graphics Interface*, 1988.

[16] S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, and M. Gross. Unified simulation of elastic rods, shells, and solids. *ACM Transactions on Graphics (TOG)*, 29(4):39, 2010.

[17] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross. Example-based elastic materials. In *ACM Transactions on Graphics (TOG)*, volume 30, page 72. ACM, 2011.

[18] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. In *ACM SIGGRAPH 2005 Courses*, page 4. ACM, 2005.

[19] M. Müller and M. Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, pages 239–246. Canadian Human-Computer Communications Society, 2004.

[20] M. Muller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[21] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151. Eurographics Association, 2004.

[22] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. In *Computer Graphics Forum*, volume 25, pages 809–836. Wiley Online Library, 2006.

[23] M. Nesme, P. Kry, L. Jeřábková, and F. Faure. Preserving topology and elasticity for embedded deformable models. In *ACM Transactions on Graphics (TOG)*, volume 28, page 52. ACM, 2009.

[24] D. Nguyen, R. Fedkiw, and H. Jensen. Physically based modeling and animation of fire. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 721–728. ACM, 2002.

[25] J. O'Brien, A. Bargteil, and J. Hodgins. Graphical modeling and animation of ductile fracture. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 291–294. ACM, 2002.

[26] J. O'Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM Press/Addison-Wesley Publishing Co., 1999.

[27] D. Pai, K. Doel, D. James, J. Lang, J. Lloyd, J. Richmond, and S. Yau. Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 87–96. ACM, 2001.

[28] G. Picinbono, H. Delingette, and N. Ayache. Nonlinear and anisotropic elastic soft tissue models for medical simulation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1370–1375. IEEE, 2001.

[29] W. Rungjiratananon, Y. Kanamori, and T. Nishita. Elastic rod simulation by chain shape matching with twisting effect. In *ACM SIGGRAPH ASIA 2010 Sketches*, page 27. ACM, 2010.

[30] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 81–90. Eurographics Association, 2007.

[31] J. Spillmann and M. Teschner. C o r d e: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 63–72. Eurographics Association, 2007.

[32] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214. ACM, 1987.

[33] C. Twigg and Z. Kačić-Alesić. Point cloud glue: constraining simulations using the procrustes transform. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 45–54. Eurographics Association, 2010.