A New 3D Spring for Deformable Object Animation

Il-Kwon Jeong, Inho Lee

Virtual Reality Department, Computer & Software Research Laboratory
Electronics and Telecommunications Research Institute, Taejon, Korea
jik@etri.re.kr

Abstract

A new3D spring for deformable object animation is proposed. Currently mass-spring system is most widely used in a deformable object animation as well as cloth animation. In order to perform a realistic cloth animation using the conventional mass-spring system, one requires three kinds of springs, that is, structural spring, shear spring, and bend spring. Performing a deformable object animation of a given geometric model also requires user's knowledge or trial-and-error on constructing a stable spring network. For example, a cube model constructed from the edges used in faces for rendering would collapse, and one should use additional springs connecting the vertices inside the cube in order to make a stable spring model for simulation. Using the proposed 3D spring all this tedious and complicated modeling procedure can be omitted and one can easily construct a stable mass and 3D spring model from a geometric model with arbitrary shape. In addition to that, our 3D spring makes it possible to animate sharp folds of a very thin object easily and efficiently without any additional geometrical procedure.

Keywords:

3D Spring, Deformable Object Animation, Physically Based Simulation

1. Introduction

The computer graphics community became interested in cloth (deformable object) modeling in the late 1980s while the beginning of modern cloth modeling research is traced back to the 1930s. The computer graphics community began to show great interest in modeling cloth and cloth structures for use in computer-generated images and animation [1].

There are increasing demands for producing physically valid animations than ever owing to the advances in computing hardware and lowered computational burden. Physically based simulation engine is a good example for such a trend. Physically based simulation can add realism to graphic content and make a user interact with the content in real-time and in a realistic way [2].

Some of commercial physically based motion engines provide deformable object simulation using the mass-spring systems. Our concern is about the dynamic simulation of deformable objects. Conventional mass-spring method requires user's intervention in constructing a stable simulation model, so it is hard to generate a stable model from a geometric model automatically.

In this paper, we propose a new 3D spring model for deformable object animation, which is not computationally heavy. The proposed 3D spring can be thought as an extension or generalization to the conventional spring. Our 3D spring exerts a rotational force as well as linear expanding or shrinking force. This feature enables one to animate a sharp wrinkle of a very thin object easily, and construct a stable spring model directly from a geometric model without tedious and complicated modeling procedure to ensure the stability.

2. Previous Work

The computer graphics community have been especially interested in animating cloth like objects among the various deformable objects. Previous works can be categorized as two approaches. They are geometric techniques and dynamic simulation. Because our concern is the latter, we will discuss the papers that describe the dynamic simulation for deformable objects or cloth.

Some of recent researches have focused on the real-time aspect. Baraff et al. used the implicit Euler integration

method such that large time steps can be used [3]. Desbrun et al. split the forces in mass-spring system into linear and non-linear parts. The non-linear part is first neglected to rapidly approximate the implicit integration and then the estimate is corrected to preserve momentum [4]. Kang et al. improved this method with an approximate implicit integration and they also improved the appearance of the cloth by tessellating the triangle and introducing a wrinkled cubic spline curve [5]. Choi and Ko proposed a semi-implicit cloth simulation technique that is very stable and responsive to produce a very realistic-looking cloth [6]. However, all these methods work on a conventional mass-spring system. So the demerits of the conventional system that are described in the next subsection still remain.

Sederberg and Parry introduced the use of free-form deformations(FFDs) as an efficient method for aimating soft bodies via a structural hyperpatch [7]. An FFD works by positioning a 4x4 lattice of control vertices(CVs) around the model. Lander proposed an idea that constructing a mass-spring system for those CVs, that is, building a jelly cube model with 16(=4x4) mass points [8]. However, this approach is not good for an interactive simulation because it is hard to define the realationship between an exerted external force and the CVs due to the ambiguity in matching between the CVs and the vertex of model being affected by the force.

Müller et al. proposed a stiffness warping technique for real-time deformation based on Green's non-linear strain tensor [9]. Though they suggested a relatively fast and accurate method applicable to both FEM and mass-spring systems, the resulting algorithm is complicated due to the inherent complexity in elasticity-based methods.

Cordier and Thalmann proposed a hybrid method that use several different algorithms for the pieces of a garment for a real-time animation of a fully dressed virtual human [10]. Again this method is based on the geometric techniques and the conventional mass-spring system.

Oshita and Makinouchi presented a method combining dynamic simulation and geometric techniques [11]. However, their normal control mechanism is complicated and is not intuitive.

For shape design and rapid prototyping applications, Szeliski and Tonnesen developed oriented particles that are new distributed model of surface shpae [12]. In addition to having a position, an oriented particle also has its own local coordinate frame. To force oriented particles to group themselves into surface-like arrangements, they devised a collection of new potential functions. However, they have dealt with separated particles and not linked particles and the potential functions used are heuristic. And the algorithm is not suitable for interactive dynamic simulation.

House and Breen suggested the use of three kinds of cloth springs, separation, bending, and trellising. Because they are imaginary springs and the magnitude of each spring force or torque is determined by differentiating the corresponding energy equation of the draping model, it is far from a physically proper dynamic simulation model. And it only uses the position of the masses, so a rumple or creases in a pleat skirt cannot be animated.

2.1. Problems in Conventional Mass-Spring System

In order to animate a cloth-like object one can arrange the masses and springs in a rectangular grid. In order to emulate more cloth-like behaviour one need to include three types of springs. These are structural springs, shear springs, and bend (flexion) springs. The shear springs prevent the model from being stretched unacceptably in diagonal directions and the bend springs exert forces against a bending (folding) of the model.

Adding two extra types of springs in addition to the structural springs is not intuitive. The need for the extra types of springs originates from the linear (1 dimensional) nature of the conventional spring. Figure 1 explains this 1D property of conventional springs. Where the configuration change does not introduce any spring force.

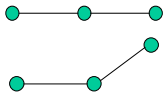

Figure 1: *configuration change of a spring model where no force is created*

Figure 2 shows an intuitive spring model for a cube-like deformable object and a collapsed cube that does not introduce any spring force.

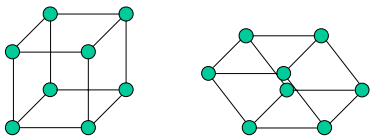

Figure 2: *an intuitive spring model for a jelly cube and a collapsed cube*

Figure 3 shows a stable arrangement for a jelly cube. It has number of diagonal springs to stabilize the model. Given a geometric model with arbitrary shape, constructing a stable spring model is not an easy task without simulating it first.

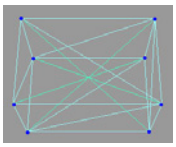

Figure 3: a *stable arrangement for a jelly cube*

3. 3D Spring

This section describes our new 3D spring model. We named our new spring model as '3D spring' because it

creates forces in additional 2D space in addition to the conventional lD space (the line coincide with the spring).

3.1. Motivation

If we can design a spring such that it creates a restoring force to its original configuration in the second situation in Fig. 2, the first model in Fig. 2 would be a valid jelly cube.

3.2. 3D Spring Model

Figure 4 shows the proposed 3D spring model. The two mass points, A and B, are connected by a 3D spring. In our spring model the mass point has its orientation as well as its postion in 3D space.

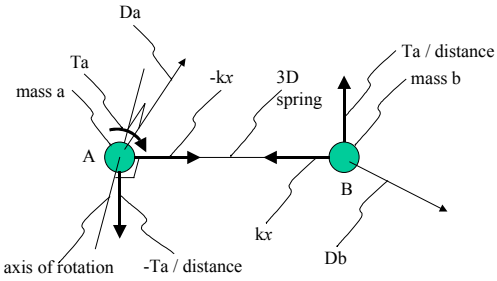

Figure 4: *3D spring model*

Our 3D spring preserves the 1D spring's characteristics and it creates additional rotational force. So it can be thought that our 3D spring is a generalization of a conventional 1D spring. Thus any previously modelled deformable object can be reconstructed using the new 3D spring model without altering the arrangement of springs. All one has to do is substituting the masses and springs with an oriented mass and 3D spring, respectively.

Each mass in Fig. 4 has a direction vector for each attached spring. This vector indicates the spring's restoring direction. A torque is created if this vector and the spring are not aligned. The torque is applied with respect to the axis of rotation that is perpendicular to the plane defined by the direction vector and the spring direction. All direction vectors are assumed to be normalized.

For the mass A in Fig. 4 the direction vector is **D**a and the torque created is **T**a. If we represent the spring direction as **vec**AB **T**a is calculated as follows.

$$\mathbf{T}a = (\mathbf{D}a \times \mathbf{vec}AB)*KTp*\theta \qquad (1)$$

Where × is a cross product operator, KTp is a torsional spring coefficient and θ is the angle between **D**a and **vec**AB with respect to the rotation of axis of mass A. Torque **T**a causes a force on mass B that is perpendicular to the plane defined by the axis of rotation of mass A and spring direction. This force is proportional to **T**a times the inverse of the distance from A to B. As a reaction of this force a force with the same amplitude and opposite direction is applied on the mass A. The forces created by mass B can be calculated in a similar way.

Listing 1 shows the pseudo code for calculating the spring forces and torques according to the above explanations. Where **V**a is the velocity vector, **P**a is the position vector, **W**a is the rotational velocity vector, and L is the spring length at rest. Parameters for mass B can be similarly defined. It also has damping coefficients, Kd for 1D spring action and KTd for torsional action. Note that **D**a and **D**b are transformed direction vectors according to the orientation of the mass A and B, respectively.

Computed torques and forces can be integrated by using conventional methods such as Euler method. It should be noted that a rotational velocity vector **W** has a meaning that the object is rotating about the axis **W** with the rotational velocity |**W**|. Using this relation we can update the orientation matrix of a mass point using a rotational transformation matrix constructed from rotation axis **W** and rotation amount |**W**|*dT, where dT is a sampling interval.

Listing 1: *pseudo code for calculating the spring forces and torques*

```
Function [Fa, Ta, Fb, Tb] = spring(Pa, Va, Da, Wa, Pb,
Vb, Db, Wb, Kp, Kd, KTp, KTd, L)

vecAB = Pb – Pa;
dist =  norm(vecAB)
vecAB = normalize(vecAB)

θ = acos (Da•vecAB)
Ta = (Da × vecAB)*KTp*θ – Wa*KTd
Fb = (Da – vecAB*(Da•vecAB)) )*KTp*θ / dist
Fa = –Fb

Fa = Fa + vecAB*(dist – L)*Kp
Fb = Fb – vecAB*(dist – L)*Kp

vecBA = – vecAB
θ = acos (Db•vecBA)
Tb = (Db × vecBA)*KTp*θ – Wb*KTd
Fa =  Fa + (Db – vecBA*(Db•vecBA)) )*KTp*θ / dist
        – damping(Kd)
Fb =  Fb – (Db – vecBA*(Db•vecBA)) )*KTp*θ / dist
        – damping(Kd)
```

4. Implementation Results and Applications

This section describes the implementation results of the proposed spring model and our on-going project as a possible direction for applying the proposed model.

Figure 5 shows results from the simulation of the jelly cube with the initial arrangement of the second model in Fig. 2. The results show that the jelly cube model in Fig. 2 suffices for a stable spring model as we intended.

Using the proposed 3D spring model we simulated a hanging cloth with creases on a table. Figure 6 (a)~(d) show creases that were impossible for the previous techniques without any geometrical procedure.

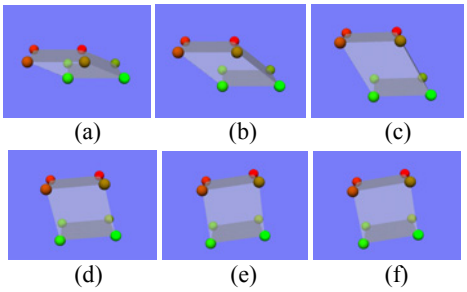

Figure 5: *implmentation results – jelly cube with 3D springs*

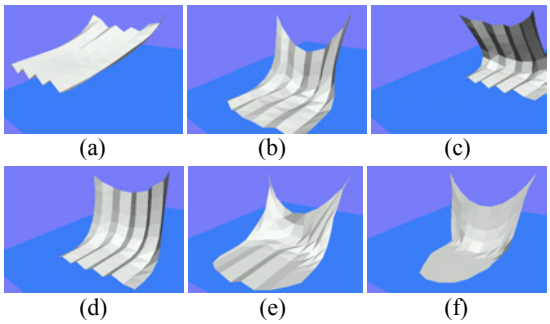

Figure 6: *implmentation results – cloth with creases: (a)~(d) – 3D spring , (e)~(f) – conventional spring*

Figure 7 shows a bar attached to a wall under gravity. The bar was modelled by stacking 10 cubes in Fig. 5.

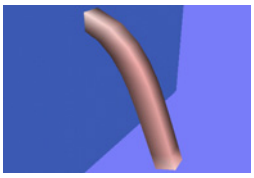

Figure 7: *a bar attached to a wall*

A brute comparison using the CPU time reveals that our 3D spring requires almost the same computation as the conventional spring in the jelly cube case (required # of springs are 12 and 20 for 3D and conventional springs).

Our on-going research includes development of physically based motion engine (PBM) (Fig. 8). Our current PBM supports rigid-body dynamics and cloth animation using the conventional mass-spring system. We are going to incorporate the 3D spring model with our PBM. In order to do so we are now implementing a collision handling procedure for our 3D spring model.

5. Conclusions

We proposed a new 3D spring for deformable object animation. One can easily construct a stable oriented mass-3D spring system from a geometric model directly.

The most beneficial feature of our 3D spring is that it can simulate a very thin objects with creases such as a pleat skirt. To our knowlege there has been no simple spring model that exhibits the above properties. Further research includes finding a possible optimization of the calculations of forces and torques.

Figure 8: *our PBM under our 3D game engine (Dream3D) – a fluctuating bridge and a waving flag*

References

1. D.H. House and D.E. Breen. *Cloth Modeling and Animation*. A K Peters, Ltd., 2000.
2. D.M. Bourg. *Physics for Game Developers*. O'reilly, 2001.
3. D. Baraff and A. Witkin. Large steps in cloth simulation. *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 98, 43-54, 1998.
4. M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. *Proc. of Graphics Interface*, 1-8, 1999.
5. Y.M. Kang, J.H. Choi, H.G. Cho, and D.H. Lee. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer*, **17**(3):147-157, 2001.
6. K.J. Choi and H.S. Ko. Stable but responsive cloth. *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 2002, 604-611, 2002.
7. T.W. Sederburg and S.R. Parry. Free-form deformation of solid geometric models. *ACM Computer Graphics (Proc.. of SIGGRAPH '86)* , **20**:151-160, 1986
8. J. Lander. Graphic content - in this corner... the crusher!. *Game Developer Magazine*, 17-22, June 2000.
9. M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. *Proc. of ACM SIGGRAPH Symposium on Computer Animation*, 49-54, 2002.
10. F. Cordier and N.Thalmann. Real-time animation of dressed virtual human. *Computer Graphics Forum (Eurographics '02 Proc.)*, **21**(3):327-335, 2002.
11. M. Oshita and A. Makinouchi. Real-time cloth simulation with sparse particles and curved faces. *Proc. of Computer Animation*, 220-227, 2001
12. R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *ACM Computer Graphics (Proc. of SIGGRAPH '92)* , **26**:185-194, 1992.