Plausible Virtual Paper for Real-time Applications

Young-Min Kang Heng-Guang Zhang
Tongmyong University
Busan, 608-711, Korea
ymkang@tu.ac.kr

Hwan-Gue Cho
Pusan National University
Busan, 609-735, KOREA
hgcho@pusan.ac.kr

Abstract

We propose an adaptive mesh animation techniques for virtual paper simulation. The proposed method can be applied to arbitrary triangular mesh structures and efficiently produces wrinkles and creases on the paper surface with stable numerical integration and deformation-based mesh refinement.

Keywords: virtual paper, physically-based modelding

1 Introduction

Since Terzopoulos et al. (1987) simulated deformable object in computer graphics literature, Baraff and Witkin (1998); Choi and Ko (2002); Meyer et al. (2001); Volino and Magnenat-Thalmann (2001) proposed various techniques for soft object animation, and Ma and Baciu (2006) devised a method to generate seams and wrinkles for realistic appearance. Grinspun et al. (2003) introduced a discrete shell model for describing thin objects. In this method, however, adaptive mesh restructuring was not taken into account. Although the method was improved for origami simulation in Burgoon et al. (2006), their work did not consider the arbitrary crumpling with external forces in interactive applications. In this paper, we propose an adaptive and stochastic mesh reconstruction method for simulating the behavior of soft and thin virtual paper objects in an interactive application. Inextensible thin objects can be represented with stiff mass-spring models, and Baraff and Witkin (1998) proposed an implicit integration scheme for stiff differential equation. In an implicit integration scheme, the mass-spring simulation can be expressed as $\Delta\mathbf{v}^{t+h} = h(\mathbf{M} - h^2\frac{\partial\mathbf{f}_\sigma}{\partial\mathbf{x}} - h\frac{\partial\mathbf{f}_\delta}{\partial\mathbf{v}})^{-1}(\mathbf{f}_\sigma^t + \mathbf{f}_\delta^t + h^2\frac{\partial\mathbf{f}_\sigma}{\partial\mathbf{x}}\mathbf{v}^t)$ where $\mathbf{x}$, $\mathbf{v}$, $\mathbf{f}_\sigma$, and $\mathbf{f}_\delta$ are the vectors of locations, velocities, spring forces, and damping forces respectively, and $\mathbf{M}$ denotes the mass matrix. We employed an approximate integration proposed in Kang and Cho (2004). However, the resulting animation does not look like paper because the static mesh structure cannot generate any crumples on the surface. Therefore, we employed an adaptive mesh techniques which was first introduced in Kang and Cho (2008).

2 Adaptive Structure with Breakable Springs

Fig.1 illustrates the breakable spring model. The compressed spring is broken into two distinct spring edges in order to maintain the original length. Two additional springs are inserted to maintain the triangular structure, and an auxiliary spring is added to preserve the damage. The total mass cannot be changed at any condition. After a new mass m_n is inserted, the neighboring masses m_0, m_1, m_2, and m_3 are adjusted as m_0', m_1', m_2', and m_3' to preserve the total mass. Let us consider two extreme cases: (a) a new particle is placed exactly on a neighboring mass point i, and (b) a neighboring mass point i is extremely far from the new mass. In the first case, the mass of the newly added particle should be $m_i/2$. In the second case, adding the particle should not decrease m_i. With this consideration, we adjust masses as follows:

$$m_l' = \left(\frac{d_l}{2\sum_{k\in N} d_k} + \frac{1}{2}\right) m_l, \quad m_n = \frac{\sum_{k\in N} m_k}{2} - \frac{\sum_{k\in N} d_k m_k}{2\sum_{k\in N} d_k} \tag{1}$$

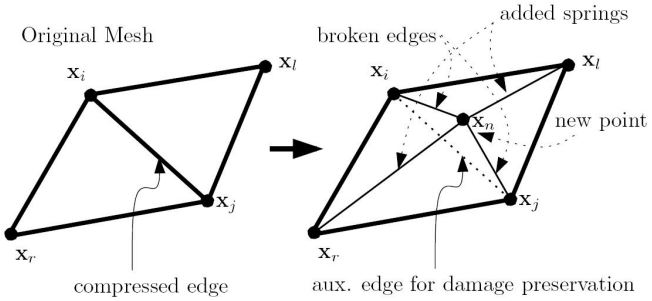


Figure 1: Breakable spring model

where d_k is the distance from the new mass to another mass point k in the undeformed configuration, and N is the set of the neighboring masses.

The momentum should be also preserved by specifying the velocity of the new mass point as follows:

$$\mathbf{v}_n = (\sum_{k \in N} m_k \mathbf{v}_k - \sum_{k \in N} m'_k \mathbf{v}_k)/m_n \tag{2}$$

3 Stochastic Edge Break Model

In our approach, the fracture of an edge occurs stochastically. Probability of the fracture is proportional to the contraction ratio $(1 - l^t_{ij}/l_{ij})$ where l^0_{ij} and l^t_{ij} denote the rest length and the current length of the spring between $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. To take the horizontal and vertical curvatures into account, we consider two neighbor vertices $\mathbf{x}_r$ and $\mathbf{x}_l$. The distance between the neighbors is l_{rl}. We denote the normal vectors at the mass-point $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_r$, $\mathbf{x}_l$ as $\mathbf{n}_i$, $\mathbf{n}_j$, $\mathbf{n}_r$, $\mathbf{n}_l$. The curvature along the edge increases the probability of the fracture while the curvature across the edge decreases it. Therefore, we can model the fracture probability to be proportional to $(1 - n_{ij})/2$ and $(1 + n_{rl})/2$ where $\mathbf{n}_{ij} = \mathbf{n}_i \cdot n_j$. In order to make it possible for a flat object to be folded, we employed a control parameter ϕ to scale the dot product of the normal vectors. Based on these observations, the probability of fracture was actually computed as follows:

$$\mathbf{P}_{ij} = \frac{1}{4l^0_{ij}}(l^0_{ij} - l^t_{ij})(1 - \phi \cdot n_{ij})(1 + \phi \cdot n_{rl}) \tag{3}$$

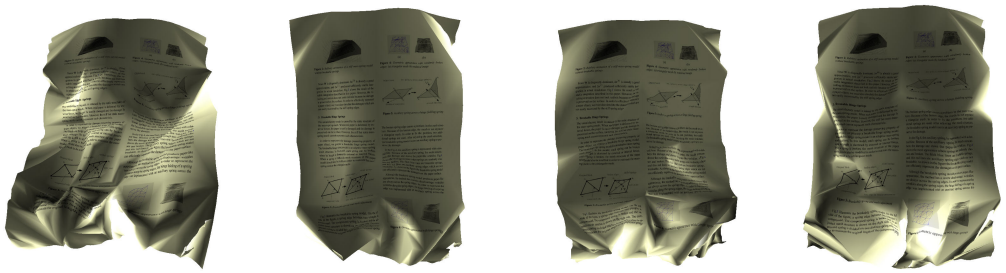
We adjust the location of the new mass-point along the surface normal vector. The magnitude of the adjustment ϑ can be easily computed as follows:

$$\vartheta = \frac{\psi}{2}\sqrt{(l^0_{ij})^2 - (l^t_{ij})^2} \tag{4}$$

where ψ is a parameter that controls length of broken edges.

4 Bending Energy based Edge Recovery

In some cases, some broken edges have to be recovered. When an edge e_{ij} is broken into two different springs e_{in} and e_{nj}, the original spring e_{ij} becomes an unbreakable auxiliary spring, i.e., $e_{ij \cdot breakable} = false$. The unbreakable auxiliary spring contains pointers to newly inserted vertex $\mathbf{x}_n$, and newly added edges e_{in}, e_{nj}, e_{rn}, and e_{nl}. When we need to remove the crumple across the broken edge e_{ij}, we can simply change the breakability property of the edge, i.e., $e_{ij \cdot breakable} \leftarrow true$, and remove one vertex $\mathbf{x}_n$ and four edges, e_{in}, e_{nj}, e_{rn}, and e_{nl}. In a more



(a) $\phi = 0.0, \psi = 1.0$ (b) $\phi = 0.7, \psi = 1.0$ (c) $\phi = 0.5, \psi = 0.0$ (d) $\phi = 0.5, \psi = 0.7$

Figure 2: Effect of control parameter ϕ and ψ: (a) $\phi = 0.0, \psi = 1.0$ (b) $\phi = 0.7, \psi = 1.0$ (c) $\phi = 0.5, \psi = 0.0$ (d) $\phi = 0.5, \psi = 0.7$

complicated situation, the broken edge e_{ij} is broken into e_{in} and e_{nj}, and the edge e_{in} or e_{nj} can be broken again. In such a case, the edge e_{ij} cannot be recovered immediately. In the proposed method, the recovery is applied to the edges which have been broken only once. Although we can recover the broken edges into an original edge, there is no criteria to determine which edges to be recovered. In our method, we exploited the bending energy of a broken edge to decide whether it would be recovered or not. A broken edge produces a crease line on the surface. The bending of the crease line recovers the broken edge. The bending energy of an edge can be conveniently computed with the model described in Grinspun et al. (2003). In their model, the bending energy $\mathbf{W}_b(e)$ of an edge e was computed as $(\theta_e - \bar{\theta_e})^2||\bar{e}||$ where θ_e and $\bar{\theta_e}$ denote the corresponding complements of the dihedral angle of the edge e measured in the deformed and undeformed configuration respectively, and $||\bar{e}||$ is the length of the edge e. In fact, the angle θ_e can be easily computed by measuring the angle between the normal vectors of two triangles incident to the edge. However, our model cannot use this angle. In order to efficiently compute the bending energy of the broken edge, we simply exploited the normal vectors at the left and right neighbors. In our model, the bending energy of a broken edge $\mathbf{W}_b(e)$ was computed as follows:

$$\mathbf{W}_b(e_{ij}) = (1 - n_{rl})/2 \tag{5}$$

where r and l denote the right and the left neighbor vertices of the edge e_{ij} respectively, and $\mathbf{n}_r$ and $\mathbf{n}_l$ are the normal vectors at those vertices.

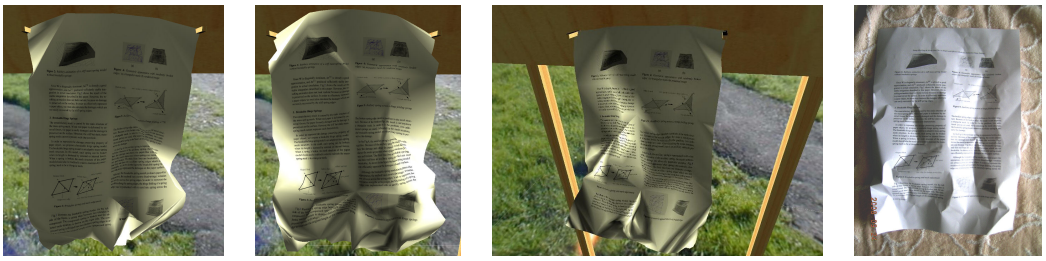
If $\mathbf{W}_b$ is larger than a given threshold ϵ, the edge is selected to be recovered. It is obvious that ϵ controls the tendency of the edge recovery. When a vertex is removed, the masses and the velocities of neighboring vertices should be adjusted to conserve the total mass and the linear momentum as follows:

$$m_i' = m_i + \frac{(\sum_{k\in N} d_k - d_i)m_n}{(|N|-1)\sum_{k\in N} d_k}, \qquad \mathbf{v}_i' = \mathbf{v}_i + \frac{(\sum_{k\in N} m_k - m_i)(m_n\mathbf{v}_n)}{(|N|-1)\sum_{k\in N} m_k} \tag{6}$$

where $|N|$ denotes the number of linked neighbors of removed mass point.

5 Experiments

Fig.2 shows the effect of the control parameters ϕ and ψ. As shown in the figure, a larger ϕ value generates stiffer paper appearance. The parameter ψ could be successfully used for controlling the crumples on the paper surface. The bending energy based strategy for the recovery of broken edges successfully works in an actual interactive animation as shown in Fig.3 (a) and (b). With a small ϵ, a virtual paper object easily recovers its broken edge. On the other hand, a large ϵ prevents the model from frequently recovering its broken edges, and it produces a crumpled surface. Fig.3 (c) and (d) compares the virtual paper generated with the proposed method and a real paper object. As shown in the figure, the proposed method plausibly reproduces the appearance of the paper object.



(a) $\epsilon = 0.1$ (b) $\epsilon = 1.5$ (c) virtual paper (d) real paper

Figure 3: The effect of ϵ and comparison with real paper: (a) $\epsilon = 0.1$, (b) $\epsilon = 1.5$, (c) a virtual paper produced with the proposed method, and (d) photo of real paper

6 Conclusion

An intuitive and efficient method for simulating virtual paper in an interactive application was proposed. The proposed method modeled the virtual paper as highly damped stiff mass-spring with breakable edges, and a stochastic edge fracture and recovery models were applied to adaptively change the mesh structure. The mass and the momentum of the adaptive mesh were preserved regardless of the structure. The experimental result shows the proposed method can efficiently control the properties of virtual paper and the result can be interactively animated in realtime applications.

Acknowledgment

This research was supported by Ministry of Knowledge and Economy, Republic of Korea, under the ITRC (Information Technology Research Center) support program supervised by IITA(Institute for Information Technology Advancement). (IITA-2009-C1090-0901-0004).

References

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pages 43–54.

Burgoon, R., Wood, Z. J., and Grinspun, E. (2006). Discrete shells origami. In *Computers and Their Applications*, pages 180–187.

Choi, K.-J. and Ko, H.-S. (2002). Stable but responsive cloth. *ACM Transactions on Graphics: Proceedings of SIGGRAPH 2002*, pages 604–611.

Grinspun, E., Hirani, A., Desbrun, M., and Schröder, P. (2003). Discrete Shells. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 62–67.

Kang, Y.-M. and Cho, H.-G. (2004). Real-time animation of complex virtual cloth with physical plausibility and numerical stability. *Presence - Teleoperators and Virtual Environments*, 13(6):668–680.

Kang, Y.-M. and Cho, H.-G. (2008). A simple and effective model for interactive paper folding. In *Poster Proceedings of Pacific Graphics 2008*.

Ma, L. and Baciu, J. H. G. (2006). Generating seams and wrinkles for virtual clothing. *Proceedings of ACM International Conference on Virtual Reality Continuum and Its Applications*, pages 14–17.

Meyer, M., Debunne, G., Desbrun, M., and Bar, A. H. (2001). Interactive animation of cloth-like objects in virtual reality. *The Journal of Visulalization and Computer Animation*, 12:1–12.

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):205–214.

Volino, P. and Magnenat-Thalmann, N. (2001). Comparing efficiency of integration methods for cloth simulation. *Proc. of Computer Graphics International 2001*, pages 265–272.