Noname manuscript No.
(will be inserted by the editor)

Shrinkage, Wrinkling and Ablation of Burning Cloth and Paper

SoHyeon Jeong · Tae-hyeong Kim · Chang-Hun Kim

Received: date / Accepted: date

Abstract The burning of a sheet of cellulose-based material, such as paper or cloth, involves uneven shrinkage which causes wrinkling. We simulate this geometrically complicated phenomenon by modeling the effects of heat transfer, shrinkage and partial ablation on a thin shell. A strain-limitation technique is applied to a two-layer structure of springs arranged as a body-centered square. Although this structure is over-constrained, convergence can be achieved using a new successive fast projection method. We also re-mesh the shells dynamically to deal with the topological changes that occur as regions burn away.

Keywords Constrained Lagrangian mechanics · Fast projection · The body-centered square · Heat transfer

1 Introduction

Heat changes the shape of solid objects by altering the material's physical and chemical state and properties. During combustion, a portion of material rapidly decomposes into gases. The residual solid is an ash which has different mass and density, causing shrinkage; as a result, the shell bends, crumples and tears, generating distinctive wrinkles.

SoHyeon Jeong
Dept. of Computer and Radio Communications Engineering, Korea University, Seoul, Korea
E-mail: SoHyeon.Jeong@gmail.com

Tae-hyeong Kim
Visual Information Processing, Korea University, Seoul, Korea
E-mail: usemagic@korea.ac.kr

Chang-Hun Kim
Dept. of Brain and Cognitive Engineering, Korea University, Seoul, Korea
E-mail: chkim@korea.ac.kr

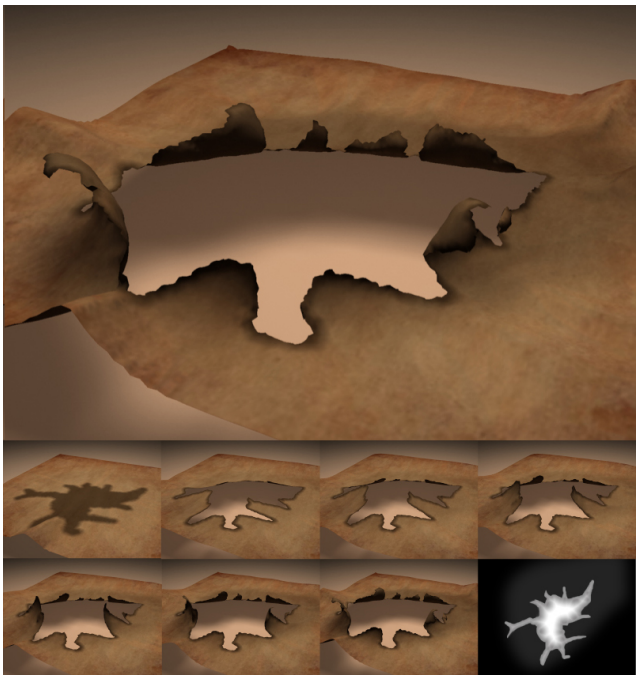

Fig. 1 Simulation of burning paper. We simulate a temperature higher than the ignition point, and weight the rate of mass loss rate using the texture shown. As the mass of the paper falls, the paper crumples and wrinkles appear.

Shells made of common combustible materials, such as cloth and paper, rarely stretch or compress in the plane of the shell, but easily bend. This property is called *developability*. When compressive forces are applied in the plane of a thin shell, it buckles but maintains its surface area. As a shell burns, mass is not lost evenly and the differential shrinkage in adjacent regions causes stresses; even though, there are no external force. But the effect is the same: the shell curves to reach equilibrium.

Recent research on the simulation of burning shells has attempted to capture these complicated deformations. Losasso et al. [15] simulated melting and burning objects by tracing the changing surface of the remaining solid region. Their results are only plausible superficially, because the deformations of burnt regions caused by shrinkage of the solid are not incorporated into their simulation. Later, Melek et al. [16] and Liu et al. [14] introduced deformation techniques to bend and crumple thin shells. However, their free-form deformation (FFD) method and the mass-spring systems produce simply curved shapes without the fine detail seen in reality, because these approaches do not fully enforce the inextensibility of thin shells.

We propose a method of simulating the bending, crumpling and wrinkling of burning shells by integrating simulations of heat transfer and the structure of developable surface. In order to produce fine wrinkles, it is necessary to model the internal dynamics of thin shells accurately. Inspired by the linear beam geometry which is used to explain the bending of shells, we propose a double-layer shell model with finite thickness based on a body-centered square configuration of springs. This structure enables us to control the stretching and bending of thin shells by using the springs to limit the strain.

This requires a robust method for determining the extensions of a network of springs. We initially tried to utilize the fast projection method. This is a powerful approach to limiting strain which is based on constrained Lagrangian mechanics. However, when applied to our proposed shell structure, fast projection diverges because the structure is overly constrained and the gradient of the spring constraints are linearly dependent. To avoid this, we break up the constraints into sets to satisfy convergence conditions and which we can project successively.

We determine the extension of each spring by considering the reduction in mass and density which occurs in regions which are burning because the temperature exceeds the ignition point. Different rates between adjacent regions, and the two layers of our shell model, buckle the shell. This approach is more physically plausible and produces more satisfyingly complicated wrinkles then existing methods, which simply relate bending to differences in temperature. To support our approach, we also provide a remeshing technique to account for the topological changes produced by burning.

We will review related work in the next section. In Section 3 we introduce our double-layer shell configuration. In Section 4, we present on the overview of heat transfer simulation process that we use to model shell deformation. In Sections 5, 6 and 7, we successively describe this heat transfer, the changes of material properties that occur during combustion and the shell dynamics. Simulation results, details of the implementations, and a discussion of the limitations of our approach follow in Section 8. Finally, we conclude in Section 9.

2 Related Work

Focusing on the generation of wrinkles in burning shells, we will briefly review work on the deformation of solid objects related to heat transfer and developable shells.

Terzopoulos et al. [21] adopted particle-spring systems to discretize 3D volumes and models phase changes in melting solids by controlling the stiffness of springs. Carlson et al. [5] simulated the melting of solids, but they adjusted viscosity within a fluid dynamics simulation. Their grid-based representation straightforwardly handles fluid flows and topological changes. Losasso et al. [15] detected the boundary of the remaining solid in burning or melting by embedding these solids within a body-centered cubic lattice and then extracting the boundary of the solid by evaluating level-set values at lattice points.

Melek et al. [16] looked at the bending and crumpling of a burning object during combustion, and presented a deformation method to simulate it in real time. The burning object is encompassed by a low-resolution grid which is subjected to free-form deformations. Liu et al. [14] also deformed thin shells indirectly by modifying an enclosing lattice. However, they modeled the crumpling forces induced by differences in temperature in lattice points on either side of the shell instead of using geometric deformation.

A realistic simulation of thin shells is feasible if its internal dynamics can be accurately modeled. Early seminal works on cloth simulation [20, 1] regarded thin shells as elastic, but an elastic shell shrinks and loses detail when compressive forces are applied in its plane. Moreover, displacements from the rest shape cause an elastic shell to produce large restoring forces which reduce numerical stability. The more accurate implicit [1], semi-implicit [4] and BDF2 [6] integration methods have been proposed to improve convergence. Nevertheless, methods based on potential energy still produce unrealistically smooth results which motivate more aggressive methods of strain limitation.

Provot [19] limited the deformation rate of springs to 10% of their rest lengths, and shrank locally elongated springs iteratively until they are within this limit. Müller et al. [18] enforced length constraints on springs by projecting two points into the line of action of a

spring until they reach valid positions. Constraint-based methods have been successfully used to simulate quasi-inextensible cloth by enforcing global constraints. Enforcing implicit constraints [13] provide more stable convergence and allows the use of a larger time-step than the explicit method. Fast projection [11] is a linearlized implicit integration which converges much more quickly than implicit methods, but still keeps strain under 0.1%. English and Bridson [9] adopted fast projection and BDF2 to maintain the rest lengths of edges in a new non-conforming discretization which prevents spurious bending forces which imparts spurious stiffness to shells.

An explicit model of bending is indispensible in the realistic simulation of wrinkles in thin shells. Early cloth simulations [1,4,12] simply controlled bending forces in terms of the angles between adjacent faces. Thomaszweski and Wacker [22] introduced a physically more accurate non-linear bending model which accounts for curvatures. Later, the linear [23], quadratic [3] and cubic polynomial [10] approximate bending models were introduced to improve performance and controllability. The bending models of Grinspun et al. [12] and Bridson et al. [4] use non-zero rest angles to preserve the curved and wrinkled rest shape of cloth while responding to collisions. The constraint-based approach [7] introduces hard angle constraints which allow a shell with sharp creases to be modeled.

3 The shell configuration

Existing cloth simulations use independent stretching and bending models because thin shells are developable. However, the stretching forces caused by uneven shrinkage in the plane of shell generate wrinkles, motivating us to control all internal dynamics using a unified model.

In this section, we first clarify the principle of bending with the linear beam element, and then explain how both bending and stretching of thin shells can be managed by in-plane tensile and compressive forces. Then we propose a particle-spring system in which a body-centered square defines two layers and finite thickness, based on the beam model.

3.1 The linear beam element

Thomaszewski and Wacker [22] explained the bending of thin shells in terms of the geometry of a simple beam (Figure 2(a)) which corresponds to the classical beam element. This beam geometry has a neutral axis (green lines), top and bottom layers (blue lines), and a finite thickness spanned by normal lines (red lines). Based on the Kirchhoff-Love Assumptions, the lengths of the neutral axis and the normal lines do not change, the whereas top and bottom layers stretch or shrink in response to the tensile and compressive stresses which occur when the neutral axis of the loaded geometry assumes a curved shape (Figure 2(c)).

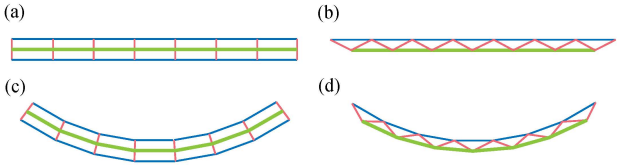

Fig. 2 (a) The linear beam geometry, and (b) and our BCS shell, shown diagrammatically in 2D. The two layers and the normals connecting them correspond to the blue, green and red springs respectively. When the top layer shrinks and the blue springs contract, (c) the beam geometry and (b) our shell both curve.

Applying the beam geometry to a shell, we see that the restoring forces in the plane of the shell correspond to bending forces exerted on the neutral axis of a beam. Stretched or compressed layers tend to be returned to their rest states by the restoring forces, the neutral axis restores flat. If a beam is extremely stiff, it tends to remain rigid, whereas flexible material such as cloth has a low stiffness. Therefore it is possible to control inextensibility and bending stiffness by applying only stretching forces to the neutral axis and the top layer (we can disregard the bottom layer, since it is dependent on the top layer).

A combined model which accounts for both stretching and bending is more appropriate for generating realistic wrinkles on thin shells for two reasons: First, maintaining the rest curvature of a stiff shell or forcing that shell to bend require large forces which make the simulation unstable. The implicit method [1] offers a solution to this problem, but implicit integration of a non-linear bending model is tricky because the gradient of the formula is difficult to calculate. Second, the bending of thin shells during burning is not caused by differences in temperature [16,14], but by shrinkage caused by changes of mass and density. The use of bending model requires an additional formula to relate the amount of shrinkage to the curvature of the shell, whereas our model bends as implicitly one layer shrinks.

A thin shell with two layers is not an approach that has been generally applied to cloth simulation, but it has already been used in [16,14] to model burning object. A thin shell can be embedded in a 3D lattice and then the lattice as deformed in response to differences in temperature between adjacent nodes in the struc-

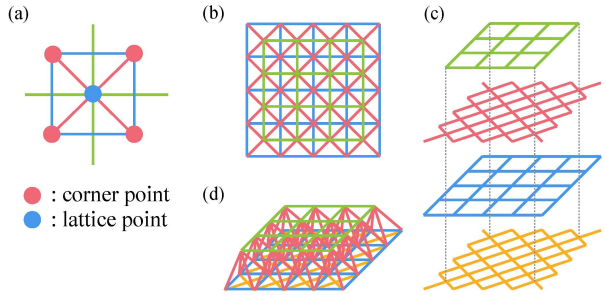

Fig. 3 The double-layer shell configuration with thickness, based on the body-centered square. The structure consists of four independent quadrilateral meshes: red, blue, green and yellow.

ture. This approach involves the assumption that the bending of thin shells is caused by the temperature difference between the top and bottom layers.

3.2 A double-layered shell based on the body-centered square

The body-centered square (BCS) is two-dimensional analog of the BCC lattice, which is commonly found in stable molecular or crystal structures. Its uniform point distribution, symmetry and stability have caused it to be used for many applications in computer graphics, including tetrehedralization [17], 3D volume deformations [17], and shell simulations [15].

The BCS consists of a set of rectangular cells, each of which has a lattice point at its center and four more points at its corners (Figure 3(a)). We can create four types of spring which connect the lattice and corner points, and we can associate each type of spring with a color: red, green, blue and yellow (Figure 3(c)), following Molino et al. [17] The green and blue orthogonal springs, $\mathbf{S}_G$ and $\mathbf{S}_B$, correspond to the top and bottom surface of the linear beam model; the red intermediate springs, $\mathbf{S}_R$, stitch the two layers together diagonally to maintain adjacency and thickness; and the yellow springs, $\mathbf{S}_Y$, link the two diagonal corner points of a unit cell to control shearing. Since the lattice points and corner points are in the same plane, the resulting spring structure is still a 2D manifold mesh (Figure 3(b)). Therefore, we offset the lattice points from the plane of the rest shell to create a shell of finite thickness (Figure 3(d)).

All the springs are used in modeling the dynamics of the shell, but only the quadrilateral mesh of blue springs is required for rendering. This mesh can easily be converted to as an isotropic triangular mesh. However, because our shell has thickness, the lattice points are no longer in the same plane as the corner points. Therefore, we replace each lattice point with a virtual center point, which is placed at the average position of the four corner points, and update its position after computing the dynamics of the shell. The resulting triangular mesh is used for collision detections as well as rendering.

4 Simulation Overview

We construct a particle-spring system using the body-centered square structure. A shell consists of a set of particles $\mathbf{P}$, and four sets of springs $\mathbf{S}_R$, $\mathbf{S}_G$, $\mathbf{S}_B$ and $\mathbf{S}_Y$. A particle p_i has physical attributes, including a position $\mathbf{x}_i$, a velocity $\mathbf{v}_i$, a mass m_i, and a temperature T_i at its position. A spring s_j connects two end particles p_0 and p_1. The length l of a spring in the rest state is called the *target length*.

At each time step n, the attributes of particles and springs are updated in the following order:

1. The temperatures $\mathbf{T}^n$ of particles is determined by simulating the heat transfer.
2. The masses $\mathbf{m}^n$ of particles at temperatures higher than the ignition point are reduced.
3. The particle-spring structure and the rendering mesh are remeshed to account for the deleted particles and springs.
4. The positions $\mathbf{x}^n$ and velocity $\mathbf{v}^n$ of particles are determined by simulating the shell dynamics and interactions with the environment.

5 Heat Transfer

A burning shell receives heat from its environment by convection and radiation, produces its own heat through combustion, and is conducted through and along the shell. Incorporating fire simulations into our system could enhance the realism of burning scenes, since changing flame shapes produce natural variations in temperature across burning shells. However, we use simpler external heat sources, such as heat balls (Figure 1 and 11) and heat textures (Figure 4). We express the change in temperature produced by these heat sources as $\mathbf{T}^{ext}$. Additionally, when the temperatures of particles exceeds the ignition point, the temperature rises move quickly due to the reduced mass and the ratio of heat to mass, μ. As a result of all this, the temperature of the particles can be expressed as follows:

$$\mathbf{T}^n = \mathbf{T}^{n-1} + \mathbf{T}^{ext} + \eta \Delta \mathbf{m}^{n-1}. \quad (1)$$

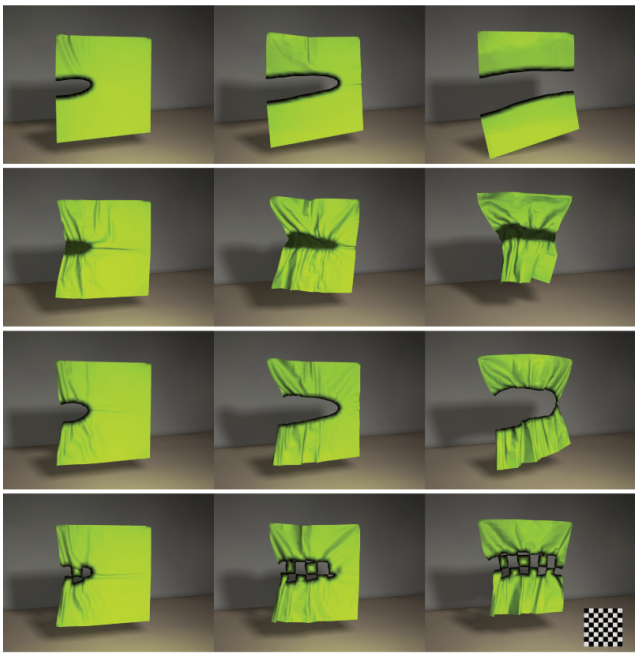

Fig. 4 Comparisons of the results of various burning effects when a very hot sphere passes across thin shells: (a) boundary change only; (b) shrinkage; (c) boundary change and shrinkage; and (d) using a texture to adjust the rate of mass loss.

We also simulate the conduction of heat over the shell. Physically accurate diffusion can be obtained by solving the heat equation.

$$\frac{\partial \mathbf{T}^n}{\partial t} = \alpha \nabla^2, \tag{2}$$

where α is thermal diffusivity.

To solve the heat equation for our particle-spring structure, we use the method of Desbrun et al.[8], which approximates the Laplacian by umbrella operators. Since the spring lengths in our model are not identical and are changed by burning, we need to consider them as parameters in the heat equation. We used a scale-dependent umbrella operator which weights the Laplacian by the inverse of the distance d_{ij} between two particles p_i and p_j.

$$L(\mathbf{x}_i) = \frac{2}{E} \sum_{j \in N_1(i)} \frac{\mathbf{x}_j - \mathbf{x}_i}{|d_{ij}|}, \quad \text{where} \quad E = \sum_{j \in N_1(i)} |d_{ij}|, \tag{3}$$

and $N_1(i)$ are the 1-ring neighbors of particle i. This enables the uniform transfer of heat across irregular meshes. By using the implicit backward Euler method we can have longer time-steps and higher thermal diffusivity, α, in a stable simulation of heat diffusion. Best of all, the diffusion of heat is unrelated to the structure of the shell, whereas the explicit method can only transfer heat from a particle to its neighbors.

6 Adjusting Target Lengths and Remeshing

A burning shell shrinks and the boundary of the remaining regions changes. We model this phenomenon by shortening springs and updating the connectivity of the particle-spring system depending on the mass change.

During combustion, the initial mass m_i^{init} of a particle i decreases by chemical processes when the temperature T_i^n is higher than the ignition point $T^{ignition}$. The reduced mass of the particle at time-step n can be expressed as follows:

$$\Delta m_i^n = -\mu h m_i^{init} (T_i^n - T^{ignition}), \tag{4}$$

where μ is the rate at which the mass decreases.

The mass at which a particle starts to disappear is m^{loss}, and m^{min} as the minimum mass that a particle can have. When m^{loss} is close to m^{init}, then the shell burns out quickly; but if m_{loss} is less than m^{min}, the shell wrinkles but remains intact. If the initial masses of two particles p_0 and p_1 connected by a spring j are m_0 and m_1, and their relative masses during burning are $m_0' = m_0 - m^{loss}$ and $m_1' = m_0 - m^{loss}$, then we can clarify the states of their connecting spring as follows:

1. Shrunk : $m_0' \geq 0$ and $m_1' \geq 0$
2. Partly burnt : $m_0' m_1' < 0$
3. Burnt : $m_0' <$ and $m_1' < 0$

The shrinkage of a spring is proportional to the change in mass and density of the two particles which it connects. Since volume is equal to mass over density, the volume of a particle p_i changes from m_i^{init}/ρ_i^{init} to m_i/ρ_i, where ρ_i^{init} and ρ_i are the densities of the initial and burnt material. We can now calculate the proportional loss in volume of the ratio of a particle v_i as follows:

$$v_i = \frac{m_i/\rho_i}{m_i^{init}/\rho_i^{init}} \quad (0 \leq v_i \leq 1). \tag{5}$$

If the proportional loss in volume of the two particles at the ends of spring j are v_0 and v_1, and l_j^{init} is the initial length, of the spring, then its new target length l_j is determined as follows:

$$l_j = l_j^{rest}(v_0 + v_1)/2. \tag{6}$$

When particles are completely burnt out ($m_i' < 0$), they can be removed from the structure, and then we remesh the particle-spring system. Since the abrupt disappearance of particles would lead to aliasing during the rendering of our lattice-based shell, we interpolate smoothly varying boundaries for the remaining regions using the technique proposed by Losasso et al. [15]: level-set values are maintained at lattice points and the structure is remeshed using marching triangles.

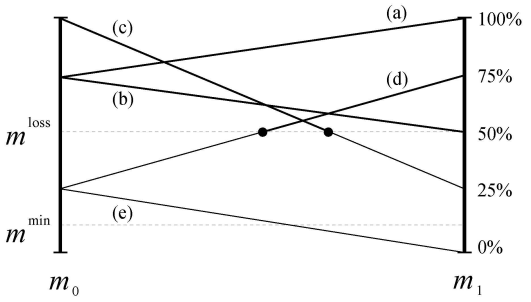

Fig. 5 The states of springs with respect to m'_0 and m'_1.

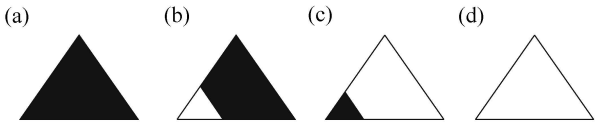

Fig. 6 Four cases of a triangle in marching triangles, depending on the unburnt mass.

A partly burnt spring ($m'_0 m'_1 < 0$) is divided into burnt and unburnt regions by an intermediate point $\mathbf{x}_{mid}$, determined by linear interpolation between $\mathbf{x}_0$ and $\mathbf{x}_1$:

$$\mathbf{x}_{mid} = \mathbf{x}_0(1-t) + \mathbf{x}_1 t, \quad \text{where} \quad t = \frac{m'_0}{m'_0 - m'_1}. \tag{7}$$

Figure 5 presents five examples which show how the state of a spring is changed by the masses of two particles that it converts. In cases (a) and (b), the spring only shrinks since both m'_0 and m'_1 are positive, and the proportional changes in length are 87.5% and 62.5%. The springs in (c) and (d) have already been shorten as much as 57.5% and 50% of their initial lengths, because they are partly burnt. In case (e), the spring is burnt out.

When the particles and springs have been updated, we remesh the particle-spring structure and the rendering mesh. Each triangle of the rendering mesh has one of four states, depending on the number of the particles remaining. A triangle is left in the rendering mesh if at least one of the particles that it connects positive relative mass, and then we remesh the remaining particles using marching triangles. If both triangles that share a spring disappear, we remove that spring from the system.

7 Shell Dynamics

Given the positions $\mathbf{x}^n$ and velocities $\mathbf{v}^n$ of particles at the beginning of time-step n, we first integrate the external forces $\mathbf{F}^{ext}$, such as gravity, using the forward Euler method:

$$\tilde{\mathbf{v}}^n = \mathbf{v}^n + h\mathbf{F}^{ext} \tag{8}$$

$$\tilde{\mathbf{x}}^n = \mathbf{x}^n + \tilde{\mathbf{v}}^n. \tag{9}$$

Since our simulation of the internal dynamics, stretching, shearing and bending of shell relies on the strain in the particle-spring systems, adopting a proper strain limiting method is critical. We prefer the fast projection method [11] based on constrained Lagrangian mechanics, because it converges quickly to the target length.

If $\mathbf{x}_a$ and $\mathbf{x}_b$ are the two end-points of a spring and l is its target length, then a constraint that preserves the spring length and its gradient can be expressed as follows:

$$C(\mathbf{x}_a, \mathbf{x}_b) = \frac{||\mathbf{x}_a - \mathbf{x}_b||^2}{l} - l = 0 \tag{10}$$

$$\nabla C_{\mathbf{x}_a}(\mathbf{x}_a, \mathbf{x}_b) = 2(\mathbf{x}_a - \mathbf{x}_b)/l. \tag{11}$$

Spring lengths that are changed by heat transfer may violate the spring constraints. Then the unconstrained positions $\tilde{\mathbf{x}}^n$ are moved to the positions $\hat{\mathbf{x}}^n$, which satisfy the constraint, by fast projection (this is explained in more detail in Section 7.1). The constrained velocity $\hat{\mathbf{v}}^n$ is obtained from the displacements during time-step h.:

$$\hat{\mathbf{x}}^n = \text{fast\_projection}(\tilde{\mathbf{x}}^n) \tag{12}$$

$$\hat{\mathbf{v}}^n = \tilde{\mathbf{v}}^n + (\hat{\mathbf{x}}^n - \tilde{\mathbf{x}}^n)/h. \tag{13}$$

Finally we correct the positions and velocities to avoid intersections with other objects and self-collision to obtain the final results:

$$\mathbf{x}^{n+1} = \text{position\_correction}(\hat{\mathbf{x}}^n) \tag{14}$$

$$\mathbf{v}^{n+1} = \text{velocity\_correction}(\hat{\mathbf{v}}^n). \tag{15}$$

Performing collision detection after fast projection could violate the constraints. But if the order of these processes is reverted, the shell could intersect other objects. Collisions with simple primitivies: such as a sphere or plane, are detected at each projection step, and the positions and velocities of particles are corrected to avoid the collision.

7.1 Successive Fast Projection

Fast projection [11] progressively projects points on to the closer manifold until they approach a constraint manifold $\mathbf{C}(\mathbf{x}) = 0$ within some threshold. At each projection step j, the unconstrained point $\mathbf{x}_j^n$ is moved by Newton's method in the direction of negative gradient of the constraint, which is expressed by the Lagrange

multiplier $-\nabla\mathbf{C}(\mathbf{x}_j^n)\lambda_{j+1}$. The position of $\mathbf{x}_j^{n+1}$ after a projection step j is determined by solving the following linear system with respect to λ_{j+1}:

$$\left(\nabla\mathbf{C}(\mathbf{x}_j)\nabla\mathbf{C}(\mathbf{x}_j)^T\right)\lambda_{j+1} = \mathbf{C}(\mathbf{x}_j). \tag{16}$$

Each displacement is then updated in turn:

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n - \nabla\mathbf{C}(\mathbf{x}_j^n)\lambda_{j+1}. \tag{17}$$

We represent this projection process at step j as a function of $\mathbf{S}$, which is the set of springs on to which we wish to enforce the constraint:

$$\mathbf{x}_{j+1}^n = \text{project}(\mathbf{x}_j^n, \mathbf{S}). \tag{18}$$

The numerical stability of the fast projection method mainly depends on the characteristics of the linear system (16). To guarantee convergence, the matrix in Eq.(16), a multiplication of the constraint gradient matrix and its transpose matrix, should be positive definite. The system will converge stably if the constraint gradients are linearly independent and the matrix is full-ranked. The ratio of the number of constraints to the number of positional DOFs of the particles is an additional concern. A ratio of more than 1 prevents a solution. For example, the average ratios of a quadrilateral or triangular mesh are 2/3 or 1, making these structures stable.

Unfortunately, the BCS does not meet these two requirements. First, the constraint gradients of the diagonal springs are linear combinations of those of the orthogonal springs which share the same particle; and so, the system matrix suffers from rank deficiency. In addition, if N_c is the total number of cells, the average sizes of the four constraint sets, $\mathbf{S}_R$, $\mathbf{S}_G$, $\mathbf{S}_B$ and $\mathbf{S}_Y$, are $4N_c$, $2N_c$, $2N_c$ and $2N_c$; and the number of DOFs of corner and lattice particles are $3N_c$ and $3N_c$ respectively. This makes the ratio of the number of constraints to the number of positional DOFs is 10:6, which is more than 1.

Our solution is to divide the constraints into subsets, each of which satisfies the convergence conditions, and solve them sequentially. We can create appropriate subsets by taking the springs of each color; the springs of each type constitute a network with the same topology as the orthogonal quadrilateral mesh used in previous techniques [11,9]. There are four spring colors, giving us four constraints to apply successively. We achieve this using a fast projection step j consisting of four subsequences. The points $\mathbf{x}_j(=\mathbf{x}_j^n)$ input to the projection step j, are moved closer to the constraint $\mathbf{C}^R$ of the spring set $\mathbf{S}^R$. The resulting points $\mathbf{x}_j^R$ are then projected on to the manifold $\mathbf{C}^G$. This is the first of four projections:

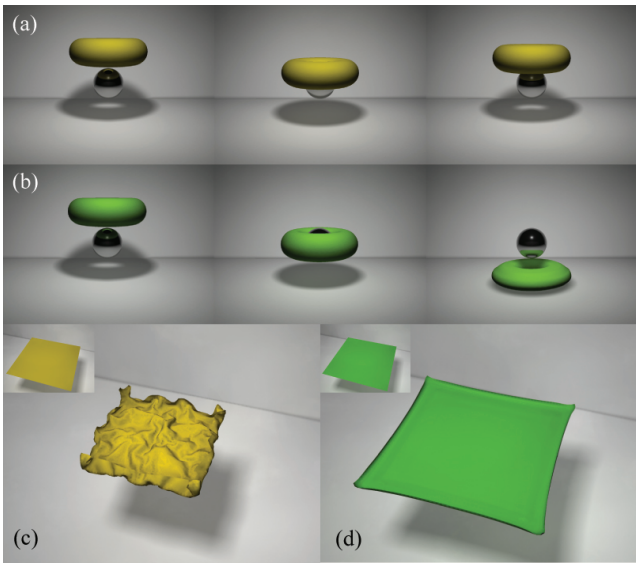

Fig. 7 Our robust strain-limiting method preserves the rest shape of thin shells and the details of wrinkles produced by shrinkage. In (a) and (b), a torus has fallen on a sphere whose radius is 1.2 times bigger than hole in the torus. In (b), the torus simulated by the mass-spring system passes the sphere and contacts ground plane since it does not preserve the target length. In (c) and (d), the whole thin shell shrinks uniformly, but only the shell simulated by our method (c) creates complicated wrinkles, whereas the shell in (d) loses its details.

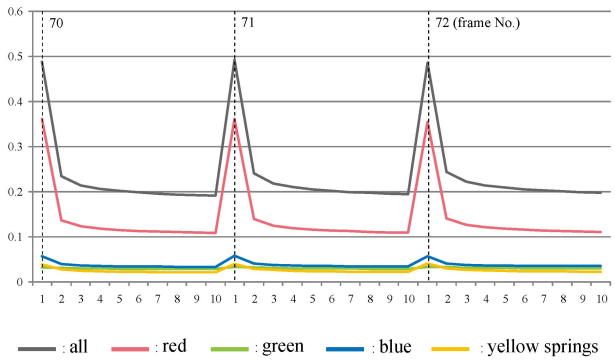

Fig. 8 Errors in $||\mathbf{C}(\mathbf{x})||$ of all types of spring and their sum. The total error (gray curve) decreases during successive fast projection.

$$\begin{aligned}
\mathbf{x}_j^R &= \text{project}(\mathbf{x}_j, \mathbf{S}^R)\\
\mathbf{x}_j^G &= \text{project}(\mathbf{x}_j^R, \mathbf{S}^G)\\
\mathbf{x}_j^B &= \text{project}(\mathbf{x}_j^G, \mathbf{S}^B)\\
\mathbf{x}_{j+1} &= \text{project}(\mathbf{x}_j^B, \mathbf{S}^Y)
\end{aligned}$$

This completes projection step j. The graphs in Figure 8 show how the constraint errors $||\mathbf{C}(\mathbf{x})||$ change through successive projection steps proceed. We see that the projection of one constraint set may increase the error of the other sets, but the sum of all the constraints always decrease.

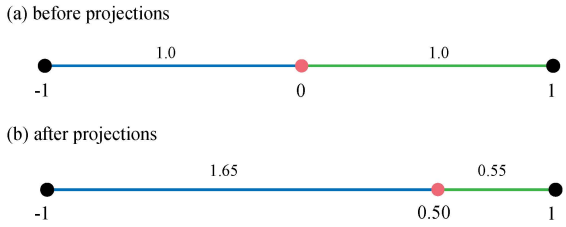

Fig. 9 Convergence of two mutually exclusive springs which share a point.

Table 1 Positions of the unconstrained particle at each projection step j.

projection step j	project($\mathbf{S}^G$)	project($\mathbf{S}^B$)
input	0.0	0.0
1	0.4306	0.4401
2	0.5527	0.4954
3	0.5767	0.5038
4	0.5804	0.5050
5	0.5810	0.5052 (output)

We experimented to see whether a point would converge at a stable location if two independent spring constraints are applied to it. Figure 9(a) shows how two types of springs, green and blue, are connected to the particle p, and the other points are fixed. When the target lengths of the two springs are changed from 1 to 1.65 and 0.55, they compete to assume their new lengths. Vibrations occurs, but the system quickly converges to an equilibrium point, as shown in Figure 9(b).

7.2 Weighted constraints

The stiffness of an elastic body depends on its material properties. Even though the constraint C($\mathbf{x}$) does not use the material stiffness as a parameter, we can obtain the same effect by relaxing the spring constraints.

Each projection in Eq.(18) is equivalent to a step in Newton's method. As the slope of the gradient $\nabla C(\mathbf{x}_j)$, increases, the variable $\mathbf{x}_j$ converges more slowly. If w is the weight applied to a spring, we divide the displacement by the spring weight to produce a weighted displacement $-\nabla\mathbf{C}(\mathbf{x}_j^n)\lambda_{j+1}/w$ which leads to the modified linear system:

$$\left(\nabla\mathbf{C}(\mathbf{x}_j)\nabla\mathbf{C}(\mathbf{x}_j)^T\right)\lambda_{j+1} = \mathbf{W}\mathbf{C}(\mathbf{x}_j), \tag{19}$$

where $\mathbf{W}$ is a diagonal matrix which contains the weights of all springs. Figure 10 shows how we can adjust the weights of the blue and yellow springs to obtain a flexible shell. It is also possible to obtain a shell which has inhomogeneous bending stiffness by assigning different weights to the springs in each subset.

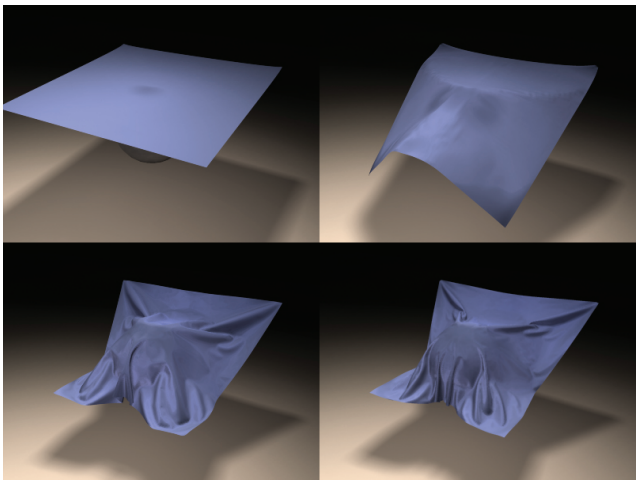

Fig. 10 Different types of thin shells produced by adjusting the weights of springs and the thickness of the shell. The ratio of the thickness of each shell to the length of its longest side and the weight for blue and yellow springs are (a) 1:20 and 1.0, (b)1:80 and 1.0, (c) 1:400 and 1.0 and (d) 1:400 and 0.1.

8 Results

We tested our shell structure using external heat sources. We assigned an initial temperature to a shell (Figure 1) and then introduced moving spheres of high temperature (Figure 4 and 11). The burning process and its randomness were controlled by adjusting the rate of mass loss μ, and the thermal diffusivity α, by applying a texture. We used the texture colors $c \in [0, 1]$ at the uv coordinates to weight the simulation parameters of each particle. The textures we used can be found with the simulation results.

Figure 1 shows a simulation of burning paper. We set the initial temperature higher than the ignition point of the paper, and varied the rate of mass loss using the shown texture. The partly burnt regions bend because of the loss of mass, but interior regions where the initial mass still remains also wrinkle due to the bending of the boundaries. In Figure 4, a hot sphere is passed across a thin shell whose two corners are fixed. As a result, the shell separates into two parts. We simulated the same scene with the addition of topological changes and shrinkage. Figure 4(a) shows a shell which only changes its connectivity [15]. As we apply more effects, the simulation results become more detailed and realistic. Figure 4(d), shows the results of changing the rate of mass loss μ using a texture to produce a tearing effect. A BCS based shell can be derived from any quadrilateral mesh: in Figure 11, we simulated heat transfer and shrinkage of a toroidal shell.

We verified our strain-limiting method by comparing results produced by a mass-spring model, based on Hooke's law, with those from successive fast pro-

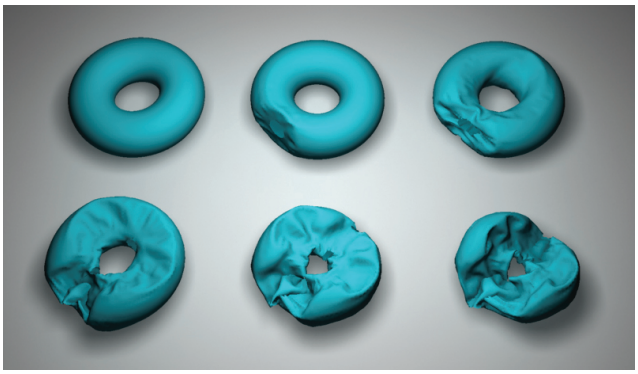

Fig. 11 Snapshots of an animation of a burning torus.

jection. Figure 9(a) and (b) show a torus falling on a sphere. The inner diameter of the torus is smaller than the sphere, and so the torus is supposed to remain on the sphere. However, the torus in Figure 9(b) has passed over the sphere and fallen to the ground, because the mass-spring model cannot preserve the strains of springs, and allows the torus to stretch. Conversely, the torus simulated by our method maintains its rest shape and bounces against the sphere. In the second example, in Figure 9 (c) and (d), we made the blue springs of the shell shrink by reducing the masses of the corner particles uniformly. Since the blue and green springs contract at different rates, realistic wrinkles appear in our simulation result (c). However, the shell in (d) lost its details, because the springs constrained by the masses and springs simply extend or compress.

8.1 Computation times

The simulation of shell dynamics takes most of the computation time, because it involves the solution of a linear system with the conjugate gradient method at each projection step j and time-step n. The bottleneck in this solution process is the matrix-vector multiplication in the conjugate gradient method. To speed up the computation, we compressed the matrix into the ELLPACK format [2], which separates values from indices. This reduces size of the matrix from N_s^2 to $N_s N_n$, where N_s is the number of springs and N_n is the maximum number of neighbors of a spring. Our shell structure can make good use of this format, because each spring set is a quadrilateral mesh and the number of neighbors of each spring is fixed (N_s =7). Using the ELLPACK structure, the computation time is linearly proportional to the number of springs. Table 2 shows computation times against the number of springs.

Table 2 Computation times for thin shells with different numbers of springs. The time-step h, the maximum numbers of iterations, and the threshold of the fast projection are 0.002, 7 and 10^{-7}. The computation times and the number of springs are linearly proportional to each other. These tests were performed on the quad-core machine with 4GB of memory. (a) The number of springs, (b) the computation time (ms) with convergence conditions, (c) the computation time (ms) without convergence conditions (simulations fully iterate), (d) the average computation time per spring ((c)/(a)).

(a)	(b)	(c)	(c)/(a)
1,000	6.50	142.46	0.142
4,000	27.93	362.69	0.090
16,000	159.17	1268.00	0.079
25,000	211.67	1877.92	0.075
10,000	1045.02	7412.92	0.074

8.2 Limitations

Successive fast projection is based upon the premise that each spring-set is a stable quadrilateral structure. If a spring-set contains points with more than six neighbors, then the simulation does not converge. Therefore it is not straightforward to apply our method to adaptive BCS, BCC or irregular meshes.

Ashes in burnt regions are easily torn and broken. We simulate this effect by reducing mass more quickly where we want the shell to break, as shown in Figure 1. However this phenomenon is not only caused by loss of mass. When two adjacent regions shrink at different rates, the region which shrinks more slowly prevents the other region from shrinking at its own speed. This creates tensile stresses in the more quickly shrinking region. If the tensile force is larger than the strength of the material left in this region, the shell tears. However, we did not model this effect, and so our examples exhibit too many wrinkles and too few tears.

The proposed shell structure maintains its thickness using only red springs. Therefore, the rapid application of a large force will make the point on one layer move into the other layer. Since we do not explicitly correct the orientation of springs, the points in the other layer remain fixed because of the red springs producing sharp crease in the shell. This problem can be fixed by collision detection, but we do not because it helps to create complicated wrinkles on the shell.

9 Conclusions

We generate realistic fine wrinkles on a burning shell using a new shell structure and a method of strain-limitation. A two-layered shell with thickness is created with a body-centered square (BCS) structure, and

this shell will wrinkle in response to stresses produced by the change in material properties that occur during burning. During successive fast projection, we deal with subsets of the springs individually, allowing our over-constrained structure to converge. Remeshing burnt regions and setting parameters using textures improves the reality of our simulation.

Acknowledgements This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2011. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No.2010-0027656) and the Second Brain Korea 21 Project.

References

1. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 43–54. ACM, New York, NY, USA (1998)
2. Bell, N., Garland, M.: Efficient sparse matrix-vector multiplication on cuda. NVIDIA Technical Report NVR-2008-004 (2008)
3. Bergou, M., Wardetzky, M., Harmon, D., Zorin, D., Grinspun, E.: A quadratic bending model for inextensible surfaces. In: SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing, pp. 227–230. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006)
4. Bridson, R., Marino, S., Fedkiw, R.: Simulation of clothing with folds and wrinkles. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 28–36. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
5. Carlson, M., Mucha, P.J., Van Horn III, R.B., Turk, G.: Melting and flowing. In: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '02, pp. 167–174. ACM, New York, NY, USA (2002)
6. Choi, K.J., Ko, H.S.: Stable but responsive cloth. ACM Trans. Graph. **21**(3), 604–611 (2002)
7. Choi, M.H., Hong, M., Welch, S.: Modeling and simulation of sharp creases. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches, p. 95. ACM, New York, NY, USA (2004)
8. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, SIGGRAPH '99, pp. 317–324. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1999)
9. English, E., Bridson, R.: Animating developable surfaces using nonconforming elements. ACM Trans. Graph. **27**(3), 1–5 (2008)
10. Garg, A., Grinspun, E., Wardetzky, M., Zorin, D.: Cubic shells. In: SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 91–98. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2007)
11. Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M., Grinspun, E.: Efficient simulation of inextensible cloth. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, p. 49. ACM, New York, NY, USA (2007)
12. Grinspun, E., Hirani, A.N., Desbrun, M., Schröder, P.: Discrete shells. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 62–67. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
13. Hong, M., hyung Choi, M., Jung, S., Welch, S.: Effective constrained dynamic simulation using implicit constraint enforcement. In: In International Conference on Robotics and Automation, pp. 4520–4525 (2005)
14. Liu, S., Liu, Q., An, T., Sun, J., Peng, Q.: Physically based simulation of thin-shell objects' burning. Vis. Comput. **25**, 687–696 (2009)
15. Losasso, F., Irving, G., Guendelman, E., Fedkiw, R.: Melting and burning solids into liquids and gases. IEEE Transactions on Visualization and Computer Graphics **12**(3), 343–352 (2006)
16. Melek, Z., Keyser, J.: Driving object deformations from internal physical processes. In: Proceedings of the 2007 ACM symposium on Solid and physical modeling, SPM '07, pp. 51–59. ACM, New York, NY, USA (2007)
17. Molino, N., Bridson, R., Teran, J., Fedkiw, R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In: IMR, pp. 103–114 (2003)
18. Müller, M., Heidelberger, B., Hennix, M., Ratcliff, J.: Position based dynamics. J. Vis. Comun. Image Represent. **18**(2), 109–118 (2007)
19. Provot, X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In: In Graphics Interface, pp. 147–154 (1995)
20. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. In: SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp. 205–214. ACM, New York, NY, USA (1987)
21. Terzopoulos, D., Platt, J., Fleischer, K.: Heating and melting deformable models (from goop to glop). Proc. Graphics Interface '89 pp. 219–226 (1989)
22. Thomaszewski, B., Wacker, M.: Bending models for thin flexible objects. In: WSCG Short Communication Proceedings, vol. 9 (2006)
23. Volino, P., Magnenat-Thalmann, N.: Simple linear bending stiffness in particle systems. In: SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 101–105. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006)

Fig. 12 Simulation of burning paper.