A new sharp-crease bending element for folding and wrinkling surfaces and volumes

Saket Patkar*
Stanford University

Ning Jin*
Stanford University

Ronald Fedkiw*
Stanford University
Industrial Light & Magic

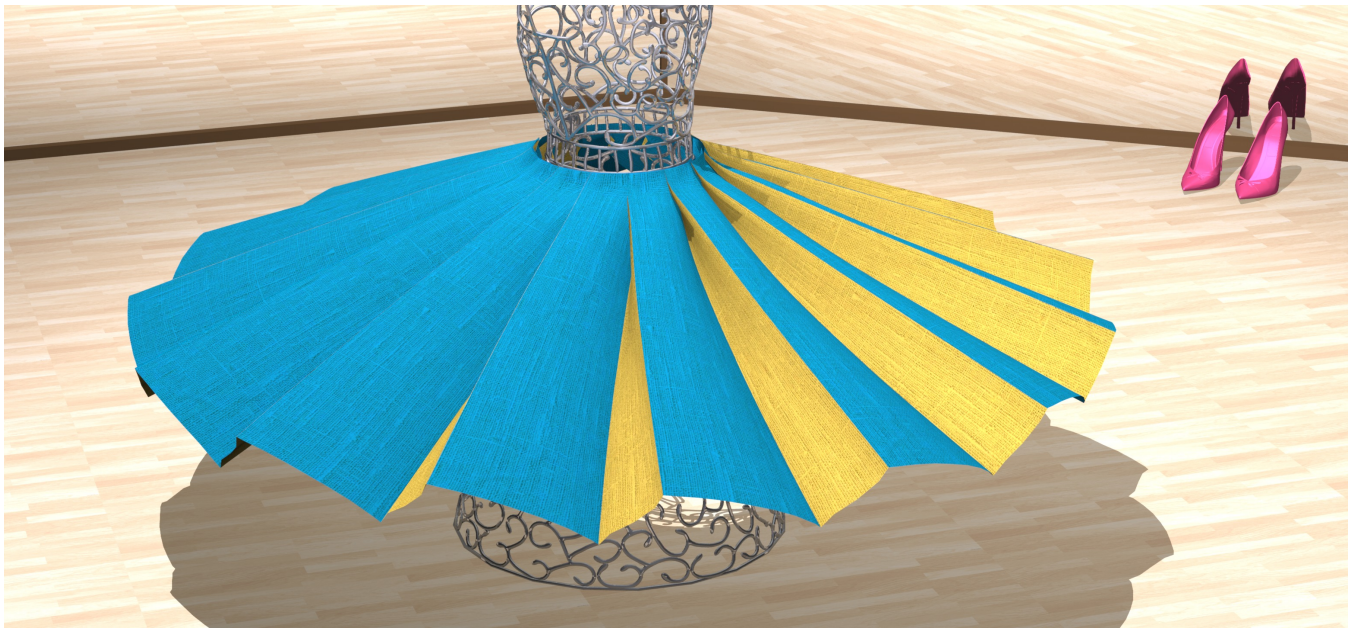

Figure 1: *A knife-pleated skirt fixed on a rotating stand rises up as the stand rotates faster. Note the sharp-creases at the pleats that are achieved by our method. (*11700 *initial triangles,* 16380 *triangles after running the virtual node algorithm)*

Abstract

We present a novel sharp-crease bending element for the folding and wrinkling of surfaces and volumes. Based on a control curve specified by an artist or derived from internal stresses of a simulation, we create a piecewise linear curve at the resolution of the computational mesh. Then, the key idea is to cut the object along the curve using the virtual node algorithm creating new degrees of freedom, while subsequently reattaching the resulting pieces eliminating the translational degrees of freedom so that adjacent pieces may only rotate or bend about the cut. Motivated by an articulated rigid body framework, we utilize the concepts of pre-stabilization and post-stabilization in order to enforce these reattachment constraints. Our cuts can be made either razor sharp or relatively smooth via the use of bending springs. Notably, our sharp-crease bending elements can not only be used to create pleats in cloth or folds in paper but also to create similar buckling in volumetric objects. We illustrate this with examples of forehead wrinkles and nasolabial folds for facial animation. Moreover, our sharp-crease bending elements require minimal extra simulation time as compared to the underlying mesh, and tend to reduce simulation times by an order of magnitude when compared to the alternative of mesh refinement.

*e-mail: {patkar,njin19,fedkiw}@cs.stanford.edu

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: folding, buckling, creasing, facial wrinkles

1 Introduction

Sharp folds and wrinkles are important features in the simulation of deformable bodies. While one can certainly achieve these effects directly from the underlying physics model, a very high resolution mesh is often required. And even in cases where the mesh resolution is sufficient, the mesh must be constructed so that its degrees of freedom align with the desired folds. Alternatively, one could create separate meshes with material on each side of the fold or wrinkle, and subsequently join or sock (see [Goulekas 2001]) them together with various ad hoc constraints. While this may work well in certain scenarios such as the boundaries between large muscles on the Hulk, it is sub-optimal for wrinkles or folds that begin and/or end organically at various places within the material such as “smile lines” (nasolabial folds) which occur within the facial fat of a character’s cheek or the wrinkles on the forehead.

Creating separate meshes is attractive when it is possible to do so because one could guarantee good mesh quality and fast simulation

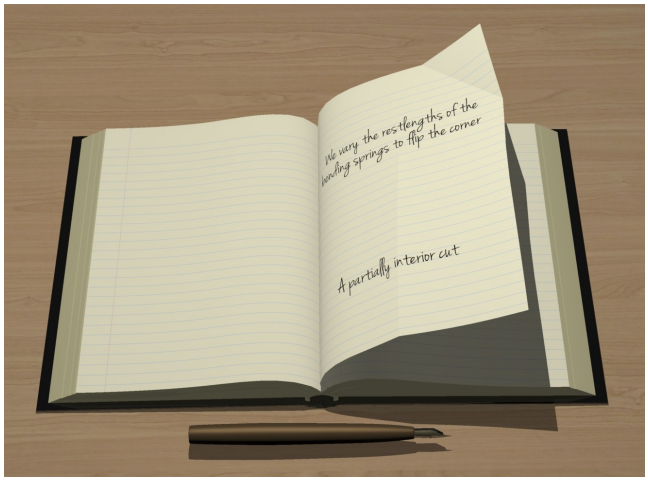

Figure 2: *Turning a creased page in a book. (*2240 *initial triangles,* 2323 *triangles after running the virtual node algorithm)*

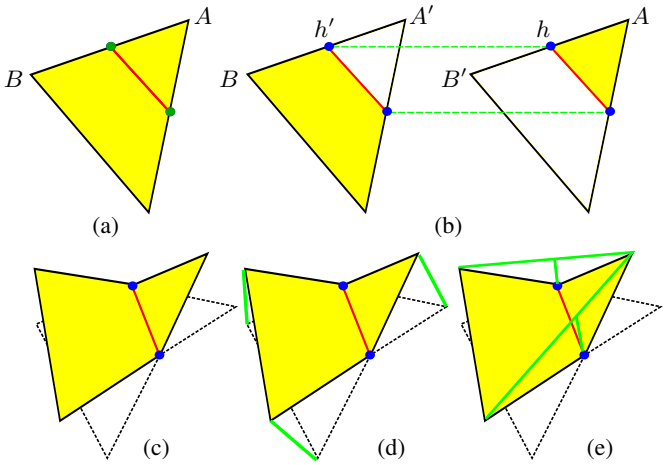

Figure 3: ***(a)*** *A triangle with an embedded segment.* ***(b)*** *The virtual node algorithm generates two triangles with embedded segments - the corresponding nodes are connected by green lines. We add hard bound particles (blue) at the embedded nodes.* ***(c)*** *Constraining the hard bound particles with their corresponding particles sews the cut together.* ***(d)*** *Bending springs between the pairs of real and virtual nodes (green).* ***(e)*** *Bending springs that extend across the cut along with axial bending springs connecting the bending spring with the hard bound particles (green).*

times a priori. This is especially salient when compared to adaptive remeshing which can create small and/or misshapen elements, further exacerbated by the need to place nodes/edges (and faces for three dimensional volumetric objects) along the fold, wrinkle, or seam. Unfortunately, it is difficult to formulate and simulate the constraints necessary to align two separate non-corresponding meshes together in a seamless fashion. Moreover, wrinkles and folds that begin and end organically within a material provide no straightforward decomposition into sub-mesh pieces. We make the novel and obvious observation that it is trivial to find correspondences between two separate meshes if one starts with a single mesh and subsequently cuts it apart into separate meshes. This also makes it easy to change the location of the fold, which is significantly more difficult when creating two separate meshes and socking them together. Moreover, making partial cuts that do not separate the mesh entirely into disjoint pieces readily allow for wrinkles and folds that originate organically within the material.

Our approach reduces the problem of creating folds and wrinkles to two well addressed sub problems, fracturing meshes and constraining points, which have both been well studied in the graphics literature. We start with a mesh with elements that are well conditioned for simulation, subsequently fracture that mesh along lines designated for folding, wrinkling, etc., and then re-constrain the corresponding nodes from different sides of the fracture back together. We fracture the mesh with [Molino et al. 2004], since it uses virtual nodes in order to avoid creating small or ill-conditioned elements ensuring fast simulation times. Although other methods such as [O'Brien and Hodgins 1999] could alternatively be used for the mesh decomposition, the creation of new smaller elements adversely reduces the size of the allowable time step. However, this could be alleviated in part by the use of a more complex implicit time integration scheme such as [Baraff and Witkin 1998].

2 Related Work

Cloth simulation has been studied in graphics for 30 years, see e.g. [Terzopoulos et al. 1987; Baraff and Witkin 1998]. We refer the reader to [Kim et al. 2013] for a survey. [Burgoon et al. 2006] presented an early approach to simulate paper folding using simple remeshing techniques. Simulating a large number of folds and wrinkles has always been a challenge, since it requires a high resolution mesh and extensive parallelism, see for e.g. [Selle et al. 2009]. One can alleviate this somewhat by using adaptive mesh refinement [Narain et al. 2012; Narain et al. 2013] to reduce the number of triangles that need to be integrated in time and collided. However, even though the number of elements is reduced, the size of the time step remains small and the condition number for the implicit system solve remains high, since there are still a number of small elements near the folds and wrinkles. There has been significant work focusing on alleviating these requirements by running a coarse simulation augmented with secondary wrinkles and folds. These are added using various techniques ranging from art direction [Cutler et al. 2007], stress and strain [Rohmer et al. 2010], relationships between coarse simulation and pre-computed fine resolution simulations [Wang et al. 2010; Kavan et al. 2011; Kim et al. 2013], secondary static simulations [Müller and Chentanez 2010], etc. [Kaufmann et al. 2009] used xFEM as an alternative to [Molino et al. 2004] for fracture to produce some sharp crease effects in shells. [Kilian et al. 2008] and [Solomon et al. 2012] addressed the problem of folding from a modeling perspective.

To produce wrinkles in volumetric meshes such as skin, [Bando et al. 2002] developed geometric models to generate fine-scale wrinkles by carving furrows and large-scale wrinkles via mesh deformation. [Magnenat-Thalmann et al. 2002] point out that in order for a volumetric material to buckle and fold it needs to be stiffer on its surface then throughout its volume. [Flynn and McCormack 2008] show that more realistic wrinkling is obtained if the skin is simulated as a three-layer model. [Rémillard and Kry 2013; Li and Kry 2014] embed a high resolution upper surface layer on a low resolution volume and use constraints to achieve wrinkles at predetermined frequencies. Studies have been done on the physical mechanism of wrinkle creation [Cerda and Mahadevan 2003], on wrinkle generation based on local muscle contraction with adaptive refinement [Zhang and Sim 2005], and wrinkle synthesis based on stress and nonlinear shell energy optimization given large scale motion [Bickel et al. 2007]. Approaches have been developed to augment a coarse input with physically simulated details using constrained Lagrangian mechanics [Bergou et al. 2007], and to use texture maps to produce the wrinkle effects [Jimenez et al. 2011]. Specifically in the context of faces, there have been efforts that use physics based models [Terzopoulos and Waters 1990; Sifakis et al. 2005; Parke and Waters 2008] to simulate the entire face in order to

achieve wrinkles and folds. Even with the increased computational power available today, these methods are too expensive to run on extremely fine resolution meshes.

3 Cut Generation and Stitching

We use triangle meshes with a simple mass-spring model for simulating cloth with linear springs along the edges, bending springs joining unshared vertices of triangles that share an edge, and axial-bending springs connecting the bending spring to the shared edge. In addition, we use the semi-implicit time integration scheme of [Bridson et al. 2003] that uses explicit integration (forward Euler in our case) for updating positions and applying elastic forces, and implicit integration for applying damping forces. However, any other simulation model including finite elements, or time integration schemes may be used with our approach.

Given a set of artist created or automatically generated input curves on the triangulated surface of the cloth mesh, we rasterize them into piecewise linear curves, with each triangle containing an embedded segment. The endpoints of these embedded segments lie on the edges of the triangle mesh and are denoted as embedded nodes - see Figure 3(a) and Figure 4 (left). We do not allow a curve to begin or end interior to a triangle and discard any portion of the curve that does, consistent with typical errors in rasterization. Furthermore, if the cut does not begin or end on the boundary of the mesh, we ensure that the cut begins and ends on a node - see Figure 4 (right). This is to ensure that the cloth has enough degrees of freedom to rotate about the cut avoiding locking that can lead to a completely flat solution. Conceptually speaking this is not a limitation of our method, but rather a constraint imposed by our use of the virtual node algorithm to achieve cutting without creating small elements that would hinder the time step or increase the number of conjugate gradients iterations. There are other related approaches such as [Wang et al. 2014; Sifakis et al. 2007a] that may be used to alleviate these restrictions, but we found our approach sufficient for the examples under consideration.

The virtual node algorithm dictates that every vertex that has a scoop cut out of its one ring donates a virtual node to all the vertices within that scoop giving them the degrees of freedom to tear apart at the cut. For the cuts allowed by our algorithm, this results in every embedded node and parent being duplicated - see Figure 3(b). Following the binding formulation from [Sifakis et al. 2007b], we add hard bound particles at the embedded nodes and note that constraining each hard bound particle with its corresponding particle on the other side of the cut sews the mesh back together - see Figure 3(c). These constraints can be achieved in various ways. For example, one could use implicit zero rest length springs; however, since springs are soft constraints the mesh may tear apart creating holes. Similarly, one could apply equal and opposite impulses to corresponding pairs of hard bound nodes, although the fact that each parent can be coupled to multiple hard bound children means many iterations would be necessary to achieve convergence. Hence we formulate the problem as a single linear system that can be solved efficiently and accurately.

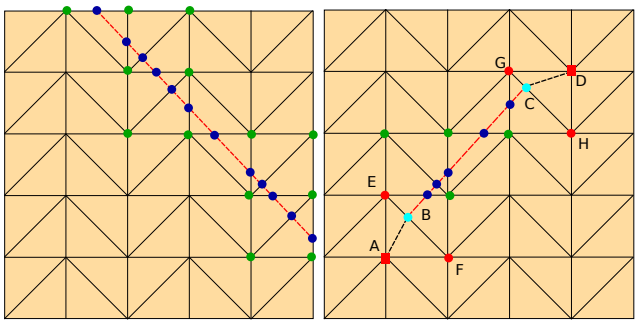

Figure 4: ***(Left)*** *A cut that starts and ends on boundaries of the mesh. The embedded nodes are shown in blue, and their parents are shown in green.* ***(Right)*** *An interior cut from point B to point C. We extend the cut to nodes A and D so that B and C become embedded nodes and E, F, G and H become parents allowing for the necessary degrees of freedom.*

Figure 5: *A character wearing a box-pleated skirt lands on the ground after completing a jump. (*21600 *initial triangles,* 31608 *triangles after running the virtual node algorithm)*

4 Post-stabilization and Pre-stabilization

We follow [Weinstein et al. 2006] to constrain pairs of hard bound particles performing a velocity based post-stabilization to zero out their relative velocities and a position based pre-stabilization to combat any drift in positions - see Figure 6. This is applied to every pair of hard bound particles (h, h') by modifying their parents (A, B') and (A', B) respectively - see Figures 3(b) and 3(c). We accomplish this by applying an impulse I_h to a hard bound particle h, and likewise $-I_h$ to particle h'. Computing the barycentric weights w_A and w_B allows us to distribute the impulse I_h to the parents as $I_A = w_A I_h$ and $I_{B'} = w_B I_h$. This changes the velocity of the parents by I_A/m_A and $I_{B'}/m_B$ resulting in a velocity change at the hard bound location of $w_A I_A/m_A + w_B I_{B'}/m_B$. The effective impulse at the hard bound location is then $I_h/m_h = w_A I_A/m_A + w_B I_{B'}/m_B = w_A^2 I_h/m_A + w_B^2 I_h/m_B$, which defines the effective mass of the hard bound particle as $1/m_h = (w_A^2/m_A + w_B^2/m_B)$. Letting $\mathbf{I_h}$ denote the vector of all impulses on the hard bound particles, $\mathbf{W}$ the matrix of weights that interpolates from hard bound particles to their parents, $\mathbf{M}$ the diagonal

mass matrix of all parents, and $\mathbf{\Delta V}$ the vector of velocity changes at all the hard bound particle locations, we have,

$$\mathbf{\Delta V} = \mathbf{W^T M^{-1} W I_h}. \quad (1)$$

This relation specifies the velocity changes at the hard bound particles given impulses acting on the hard bound particles.

In order to match the velocity of a hard bound particle and its corresponding particle we require $V_h + \Delta V_h = V_{h'} + \Delta V_{h'}$, or

$$\Delta V_h - \Delta V_{h'} = V_{h'} - V_h. \quad (2)$$

Writing Equation (2) for all such pairs of hard bound particles results in,

$$\mathbf{W_{new}^T M^{-1} W_{new} \hat{I}_h} = \mathbf{V_{h'} - V_h} \quad (3)$$

The left hand side of Equation (3) is obtained as follows. $\mathbf{I_h}$ from Equation (1) contains an entry I_h for a hard bound particle h and an equal and opposite entry $-I_h$ for its corresponding hard bound particle h' for every pair (h, h'). Therefore we can coalesce $\mathbf{W}$ into $\mathbf{W_{new}}$ by subtracting a pair of columns for every pair (h, h'). In addition, we remove the $-I_h$ entries from $\mathbf{I_h}$ to obtain $\mathbf{\hat{I}_h}$ which is half the size, and note that $\mathbf{W_{new} \hat{I}_h} = \mathbf{W I_h}$. Thus $\mathbf{M^{-1}}$ does not need to change, and $\mathbf{W_{new}^T}$ simply combines the change in velocities on the left hand side of Equation (1) to match the left hand side of Equation (2).

Generally speaking, $\mathbf{W_{new}^T M^{-1} W_{new}}$ is a symmetric positive definite (SPD) matrix, which is constant throughout the simulation unless wrinkles are being dynamically added or removed. Theoretically speaking, $\mathbf{\hat{I}_h}$ has a length equal to the number of nodes in a cut making it one dimensional, and thus the coefficient matrix is two dimensional or on the same order as the number of nodes in the mesh. However, for coarse simulations with many wrinkles, such as the pleats in Figure 1, a large number of segments may have embedded nodes on a scale approaching that of the number of segments in the mesh, potentially making the coefficient matrix effectively the size of the number of nodes squared. Thus, while it is tempting to precompute the inverse of the sparse coefficient matrix, the fact that its size can be large and that the inverse is dense makes applying the inverse during simulation inefficient. On the other hand,

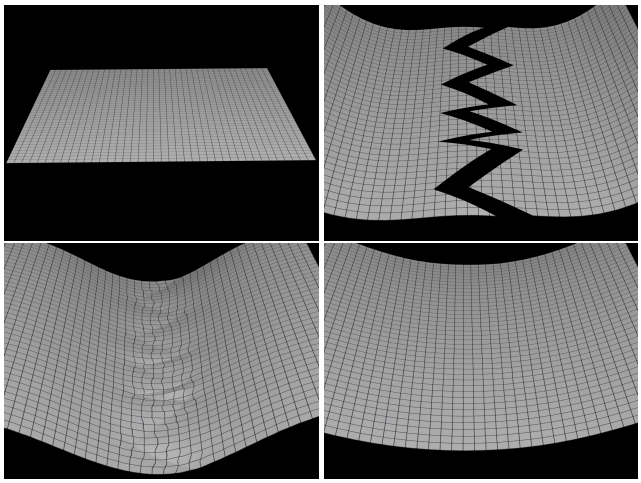

Figure 6: ***(Top left)*** *A piece of cloth rendered with a Cartesian texture map in order to visually highlight deformation and connectivity.* ***(Top right)*** *Using post-stabilization only, the cut can slowly drift apart.* ***(Bottom left)*** *Pre-stabilization eliminates drift.* ***(Bottom right)*** *Bending springs can be used to make the cut seamless, if desired.*

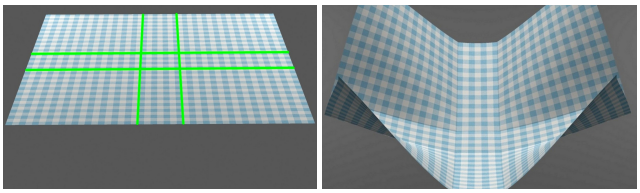

Figure 8: ***(Left)*** *A piece of cloth with the cuts (green). Embedded nodes at the intersection of the cuts have four copies after cutting the cloth with the virtual node algorithm.* ***(Right)*** *The cloth bends sharply.*

we can precompute the Cholesky factorization, which can be made rather sparse using permutation matrices which permute the order of the unknowns [Hogben 2007]. Although a forward and backward substitution is required to invert the coefficient matrix using the precomputed Cholesky factorization, it can be (and is in our simulations) significantly faster than precomputting the inverse.

The particle positions are updated using a forward Euler step: $X^{n+1} = X^n + \Delta t(V + \Delta V)$ where ΔV is the change applied to keep pairs of hard bound particles coincident in position. Although the hard bound particles are not explicitly updated but rather interpolated from their parents, we can still write the equations $X_h^{n+1} = X_h^n + \Delta t(V_h + \Delta V_h)$ and $X_{h'}^{n+1} = X_{h'}^n + \Delta t(V_{h'} + \Delta V_{h'})$ simply by linearly interpolating the position updates of the two parent particles using the barycentric weights. We desire $X_h^{n+1} = X_{h'}^{n+1}$ and thus we have

$$\Delta V_h - \Delta V_{h'} = (X_{h'}^n - X_h^n)/\Delta t + V_{h'} - V_h \quad (4)$$

analogous to Equation (2). Thus we can solve

$$\mathbf{W_{new}^T M^{-1} W_{new} \hat{I}_h} = (\mathbf{X_{h'}^n - X_h^n})/\Delta t + \mathbf{V_{h'} - V_h} \quad (5)$$

in the same manner as Equation (3) in order to guarantee that the hard bound particles stay coincident in position.

If multiple cuts intersect at a point then the embedded node at the intersection can have multiple copies (up to 4 as allowed by the virtual node algorithm) - see Figure 8. For a node with k copies, we write $k-1$ equations - one equation paring each newly added copy with the original node.

5 Collisions and Self-Collisions

We follow the collision, contact, and friction formulation of [Bridson et al. 2002] with the modifications made by [Selle et al. 2009] for parallelism, collisions, and accurate friction. For each hard bound particle location we create a new soft bound particle which is connected to the hard bound location using the binding formulation of [Sifakis et al. 2007b]. We initialize the mass of the soft bound particle to be identical to the effective mass of the hard bound particle and simulate the soft bound particle by inheriting forces from the hard bound particle. This allows the mass-spring system to oscillate with its natural frequencies carrying soft bound particles along for the ride as they track the hard bound particle locations, without the soft bound particles adversely imparting extra mass drag on the system. [Sifakis et al. 2007b] only mapped elastic forces to the soft bound particles ignoring the damping forces because of a loss of symmetry during conjugate gradients. However, we note that this can be remedied similar to [Shinar et al. 2008], by using the resulting velocities from the conjugate gradient solve in order to recompute the damping forces explicitly. When calculating the total momentum of the system, it is important to note that the soft bound particles do not contribute momentum based on their velocities, but rather based on the difference between their velocity

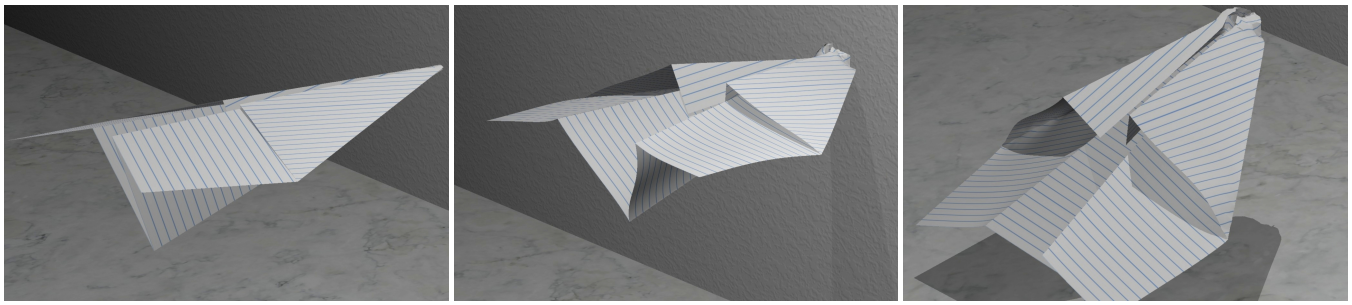

Figure 7: *A paper plane collides with a wall and deforms near the tip. Notice how the plane maintains its structure even after the impact. (*8000 *initial triangles,* 8716 *triangles after running the virtual node algorithm)*

and the velocity of the hard bound particle they are associated with. This way the system does not gain momentum as it accelerates the soft bound particles to move with the hard bound particles, but only as the soft bound particles interact via collisions etc., changing their relative velocity with the hard bound particle. On the other hand, the virtual node's momentum is directly included as part of the total momentum along with all regular cloth nodes.

We propose an impulse based fully momentum conserving alternative to the binding springs proposed in [Sifakis et al. 2007b], for managing the soft bound particles during collisions. First, soft bound particles collide with rigid bodies and otherwise interact along the lines of [Bridson et al. 2002] in a manner identical to that of regular particles in the cloth mesh (unlike virtual particles which do not for example collide). Then, in order to keep the soft bound particles from drifting too far from their hard bound target locations, we apply equal and opposite impulses to the soft bound particles and their hard bound target locations, $\pm\mathbf{I_s}$. Applying impulse $\mathbf{I_s}$ to the hard bound particles results in a change in velocity of $\mathbf{W^T M^{-1} W I_s}$, while applying $-\mathbf{I_s}$ to the soft bound particles results in a change in velocity of $-\mathbf{M_s^{-1} I_s}$ where $\mathbf{M_s}$ is a diagonal matrix of soft bound particle masses. Thus the analog of Equation (3) is,

$$(\mathbf{W^T M^{-1} W} + \mathbf{M_s^{-1}})\mathbf{I_s} = \mathbf{V_s} - \mathbf{V_h}. \quad (6)$$

where $\mathbf{W^T M^{-1} W} + \mathbf{M_s^{-1}}$ is SPD and can be solved along the lines of Equation (3). In order to synchronize positions we solve the following system similar to Equation (5),

$$(\mathbf{W^T M^{-1} W} + \mathbf{M_s^{-1}})\mathbf{I_s} = (\mathbf{X_s^n} - \mathbf{X_h^n})/\Delta t + \mathbf{V_s} - \mathbf{V_h}. \quad (7)$$

Note that every time a segment of a mesh is cut we create two embedded nodes, which are each assigned a hard bound particle location as well as a soft bound particle. We have found that applying impulses between all corresponding pairs of soft bound particles in order to sync their velocities (positions) first, before syncing them with the hard bound particles along the lines of Equation (6) (Equation (7)) provides for better results.

For self collisions we follow [Bridson et al. 2002] by generating an auxiliary mesh which will maintain a collision free state. This mesh consists of all the non-virtual particles in the original mesh along with one new particle for each pair of embedded nodes. The resulting mesh is connected in the same fashion as the original mesh but modified to include the embedded nodes and segments in the obvious way. Furthermore, all the quads are subdivided into triangles. When the simulation starts, this collision mesh is coincident with the simulation mesh and is collision free. At some later point in time a new collision free state is constructed as follows. The current simulated positions of the particles on the collision mesh are denoted as the proposed state. Then we assume linear trajectories between the last collision free state and the proposed state. Collisions and repulsions are then applied along the lines of [Bridson et al. 2002] in order to compute a new collision free state the collision mesh may evolve to. These positions are recorded as the new collision free state. The new velocity is determined by taking the velocity at the proposed state and adding in all the momentum changes resulting from the processing of collisions and repulsions using the method of [Bridson et al. 2002]. For nodes in the regular mesh we can simply add the change in momentum to that node. For nodes that correspond to the hard bound particle locations the change in momentum is distributed equally to all their corresponding soft bound particles. At this point, one could attempt to move the virtual nodes to a better location so that the hard bound particles are better synced with their corresponding positions in the collision free state. However, this is not necessary because the linear trajectories between the last collision free state and the current mesh will aim to match whatever positions the hard bound particles happen to be in, allowing (at least) for one-way syncing. On the other hand it might improve the behaviour of the simulation and/or allow for larger time steps.

6 Controlling bending

As shown in Figure 3 the distance between parent nodes and the corresponding virtual nodes increases/decreases as the rotation angle increases/decreases (i.e. A and A', B and B'). Therefore, we can add springs that connect a node with its corresponding virtual node in order to resist bending - see Figure 3(d). We use implicit

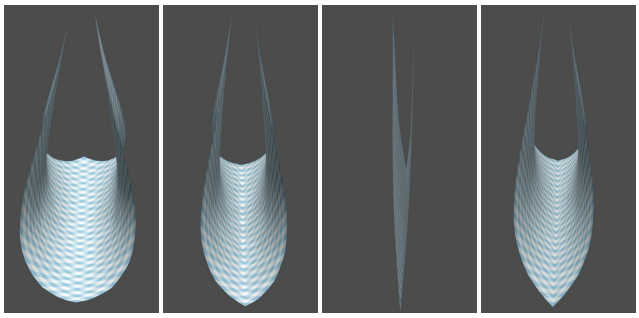

Figure 9: ***(a)*** *A piece of cloth hanging from its opposite corners.* ***(b)*** *Locally remeshing the cloth so that it has degrees of freedom on a line along the diagonal results in more bending, but makes the simulation* $10\times$ *slower (the smaller elements require a more stringent time step restriction).* ***(c)*** *Our sharp-crease bending elements can be applied with almost no additional simulation cost.* ***(d)*** *Furthermore, bending springs can be utilized to achieve results similar to remeshing, but* $10\times$ *faster.*

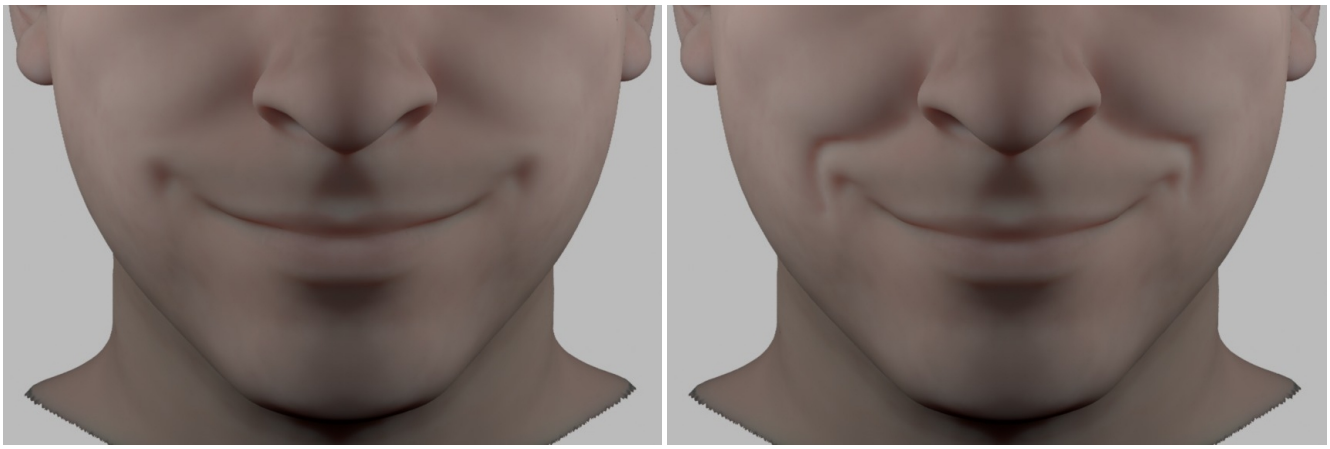

Figure 10: *We achieve a better nasolabial fold as compared to the base simulation; a very subtle effect that improves the smile dramatically. Allowing the coarse simulation mesh to fold in this fashion creates a number of subtle secondary effects in the deformation that lead to an added realism not easily captured by the addition of a secondary cloth mesh.*

zero length springs along the lines of [Sifakis et al. 2007b] so that they do not impose a time step restriction. Alternatively, one could add bending springs on edges of the material mesh that go across the cut and axial bending springs that connect these bending springs with the hard bound particles - see Figure 3(e). Note that these bending springs are at least as long as the smallest edge in the original mesh, and so they do not adversely slow down the simulation time. Equivalent results to Figure 9 (d) were achieved using both styles of springs.

For examples where we wish to begin the simulation with a fold or crease, such as the paper airplane in Figure 7, we initialize the bending springs' restlengths to their current length after making the fold/crease. Alternatively, we can create the folds by simply starting with the flat mesh and dynamically animating the restlengths of the bending springs between the real and virtual node. These springs can have arbitrarily small restlengths which can readily be handled using the non-zero restlength implicit spring formulation of [Selle et al. 2008]. Finally note that with respect to plasticity, the bending springs are treated as any other spring in the system, i.e. whenever the strain in the system exceeds the plastic yield strain we reset the restlength of the spring appropriately.

7 Folding Volumetric Objects

Given a volumetric object and a curve drawn on its surface, we first create a cutting surface by extending the cut in the normal direction towards the interior of the mesh. We extend this cutting surface about two to three layers deep into the tetrahedra. Intersecting this surface with the edges of the mesh creates triangles/quads within tetrahedra, from which the virtual node algorithm can fracture the tetrahedral mesh. The top surface of the tetrahedral mesh is then identical to a cloth/paper mesh discussed above. Thus, we can stitch it together, apply pre-stabilization and post-stabilization, apply collisions and self-collisions, etc., again, as above. Note that we do not modify the interior of the tetrahedral mesh leaving a crack underneath the surface so that the mesh has more ability to deform/overlap and/or separate leaving gaps. This further enables the interior mesh's ability to stretch and compress as stressed in [Magnenat-Thalmann et al. 2002] without requiring a weakening of the mesh. In this respect, deeper/shallower cuts in the mesh allow more/less readily for bending and folding. Although we have obtained good results using our method, if one desires even more high resolution wrinkles one could use the method of [Rémillard and Kry 2013; Li and Kry 2014]. The drawback of this approach is that it adds a high resolution cloth mesh on top of the volumetric mesh and therefore does not obtain the large displacements of the volumetric mesh that our method can. These large displacements of the primary volumetric mesh are more anatomically and physically motivated than wrinkles in a secondary skin surface, and thus have higher visual fidelity as well as non-local influence. On the other hand their method does allow one to augment our method with a higher resolution mesh to get very small scale folds and wrinkles in a multiresolution sense.

In Figure 11 we simulate a block of tetrahedra using the inverting finite volume formulation of [Teran et al. 2003; Irving et al. 2004]. Motivated by [Magnenat-Thalmann et al. 2002], we strengthen the top surface of the tetrahedral mesh so that the surface elements resist compression coaxing the object to wrinkle and fold. This is accomplished by treating it as a triangle mesh and adding additional edge and bending springs obtaining the results shown in Figure 11 (top right). One could likely obtain similar results by adding triangular finite elements or stiffening the upper layer of tetrahedra. The results obtained using our algorithm to simply cut and constrain the surface along the white lines are shown in Figure 11 (bottom left). While this does tend to induce sharp bending along the cut,

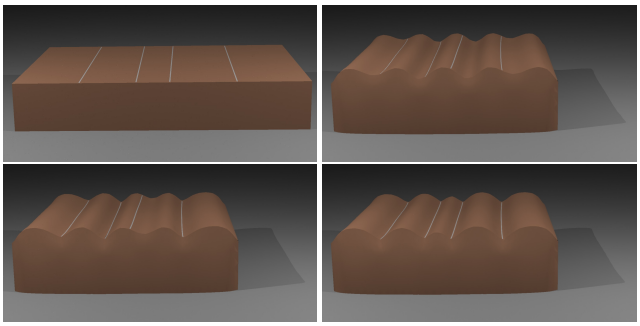

Figure 11: ***(Top left)*** *A volumetric block with a stiffer skin on top.* ***(Top right)*** *Folds form under compression.* ***(Bottom left)*** *Adding cuts along the white lines increases the likelihood of buckling at those locations.* ***(Bottom right)*** *In addition, pushing slightly downward on the hard bound particles makes the buckling exactly coincident with the cuts. Note too that the valleys are sharp, especially as compared to (Top right), which is more consistent with wrinkles and skin - see [Magnenat-Thalmann et al. 2002].*

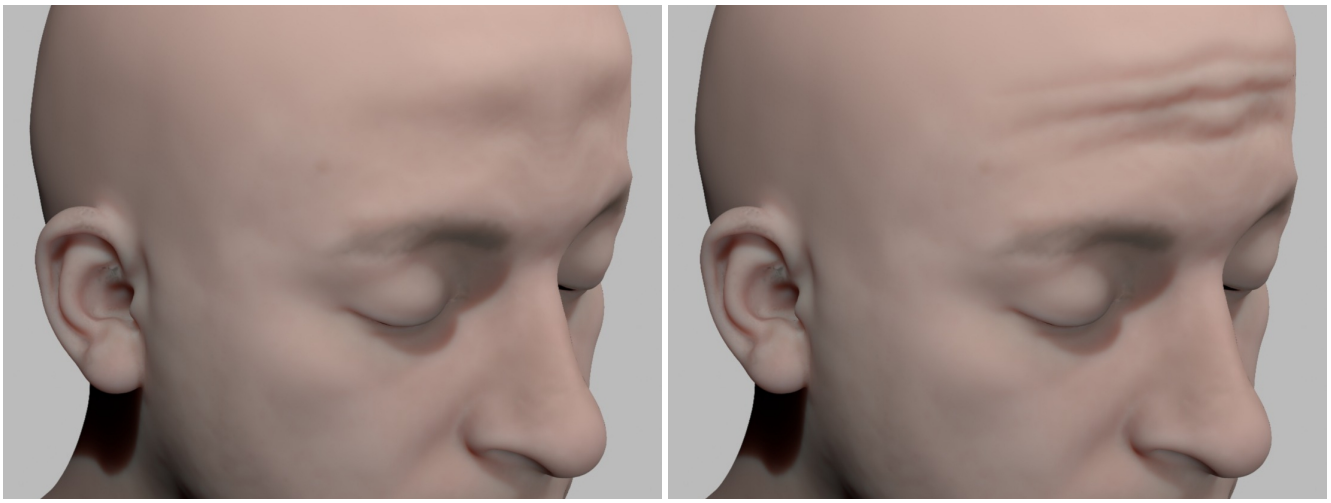

Figure 12: *We can achieve realistic and fully artist controllable wrinkles on the forehead. Although similar results could be obtained with a secondary cloth mesh, our sharp-crease bending elements allow the coarse simulation mesh to buckle resulting in other desirable secondary effects. Additionally, a secondary cloth mesh could be utilized instead for the very small high frequency wrinkles.*

some cuts only bend temporarily and may not maintain their bend especially when they are influenced by neighbouring cuts. This can be remedied by applying a small downward force on the embedded nodes to create a slight indentation very similar to wrinkles as shown in Figure 11 (bottom right).

To apply the downward force more systematically, we add anchor springs on the hard bound particles in order to pull them towards the interior of the tetrahedral mesh. Initially, the anchor spring has a specified length which indicates how deep in the normal direction the spring extends. We denote the interior end of the spring as the anchor point and store its barycentric weights with respect to the tetrahedron that contains it. We set the mass at this point to be infinite assuming that, as in our case, we are simulating flesh on top of a much heavier bone. However, it is possible to use the effective mass of a hard bound particle location within the tetrahedron and treat this as a binding. Note that one could update the position of the anchor as the mesh deforms, which might give better results for large deformations and/or high curvature, but we have found this unnecessary for our examples.

7.1 Extension to quasistatics

We follow the algorithm of [Teran et al. 2005] that uses conjugate gradients to solve for the displacement during the Newton-Raphson iterations over the positions. We modify our constraint formulation from Section 4 to handle quasistatics as follows. In this case we apply an equal and opposite push P (mass times change in position) to the pair of hard bound particles to keep them attached. Along the lines of Equation 1 we arrive at

$$\mathbf{\Delta X} = \mathbf{W^T M^{-1} W P_h}. \tag{8}$$

Since we desire $\mathbf{\Delta X_h} = \mathbf{\Delta X_{h'}}$ for post-stabilization and $\mathbf{X_h^{n+1}} = \mathbf{X_{h'}^{n+1}}$ for pre-stabilization, we obtain,

$$\mathbf{W_{new}^T M^{-1} W_{new} \hat{P}_h} = \mathbf{\Delta X_{h'}} - \mathbf{\Delta X_h} \tag{9}$$

$$\mathbf{W_{new}^T M^{-1} W_{new} \hat{P}_h} = \mathbf{X_{h'}^n} - \mathbf{X_h^n} + \mathbf{\Delta X_{h'}} - \mathbf{\Delta X_h} \tag{10}$$

similar to Equations 3 and Equations 5.

8 Examples

Figures 1 and 5 demonstrate the efficacy of our approach for simulating real-life objects such as pleated skirts. We show two popular kinds of pleated skirts: knife-pleated (one-sided) in Figure 1 and box-pleated (two-sided) in Figure 5. To make the skirt, we start with a piece of cloth that is in the shape of a sector of an annulus, and add radial cuts at a desired frequency to define the pleats. After running the virtual node algorithm and building our constraint system, we fold the mesh along the pleats by rotating the particles resulting in another sector of an annulus but with a smaller central angle. Finally, we fold this planar shape into a conical frustum and close the mesh by collapsing the particles forming the start and end of the sector. Given this conical skirt with pleats completely folded-in, we impart small radial velocities to the soft bound particles along the cuts and save the state of the cloth after the pleats have slightly separated to be used as the initial mesh in the simulation. During simulated motion, the pleats open up and fold back as expected. Even using only a simple mass-spring system, our algorithm is able to produce visual effects with the flavor of a complex mechanical system. To get an estimate of our algorithm's overhead we ran the skirt examples without our pre-stabilization and post-stabilization steps. This test was $15 - 20\%$ faster, though one should note that this range significantly overestimates our algorithm's cost. This is because without our pre- and post-stabilization steps the skirt is allowed to tear apart into small separate pieces, creating a much simpler problem.

The paper plane in Figure 7 shows that we can constrain an object to maintain a bent shape during simulation. As the plane flies and hits the wall, the impact opens up the wings, but the bending springs oppose flattening and the folded shape is largely restored. The example of a book page with a folded corner in Figure 2 illustrates the effect of the springs between real and virtual nodes that allow us to create a fold from a non-bent initial state. For both of these examples we increase the mesh size by less than 10%, and all of the newly added triangles are same size as the triangles in the original mesh. We also note that the precomputation time, which includes running the virtual node algorithm, creating the constraint system, and computing the Cholesky factorization, is quite small. For example, even with our unoptimized implementation we only need 3 seconds of precomputation for the paper plane example.

Figures 10 and 12 demonstrate the effectiveness of our method in generating facial wrinkles. For both of the examples, we drew fairly coarse curves, each containing approximately 4-6 linear segments, but still obtain convincing results. We vary the restlength of the anchor springs to fade in the wrinkles as the person smiles (flexing the zygomaticus muscle) or lifts their eyebrows (flexing the frontalis muscle). The restlengths are not only set as a function of muscle activations, but also based on their distance from the ends of the curve to enhance the organic nature of the fading in/out effect on the open boundaries of the wrinkle lines. The face model has 2.5 million tetrahedra and 35 muscles. Adding the smile lines and the forehead wrinkles increases the simulation time by 20% due to the fact that the more complex problem requires more conjugate gradient iterations to converge.

We render the collision mesh in all our examples since it always maintains a collision free state. Since the newly added vertices in the collision mesh (i.e. embedded nodes) lie on mesh edges with known interpolation weights, quantities such as texture coordinates can be easily recomputed via an interpolation from the initial mesh.

9 Conclusions, Limitations, and Future Work

We have presented a robust, art-directable, and efficient approach for generating sharp folds and wrinkles on surfaces and volumetric objects and demonstrated its potential impact through a variety of examples. A significant benefit of our approach is the computational savings in run time. Although our algorithm requires slightly more pre-computation in the initialization stage in order to set up the constraint system, the per frame simulation times are almost identical to standard simulations. Mesh refinement, on the other hand, typically significantly increases the computational effort required even when carried out adaptively.

Since the virtual node algorithm limits us to one cut per edge, we cannot make wrinkles with a frequency higher than the resolution of the mesh without some adaptive mesh refinement. In addition, the virtual node algorithm does not allow cuts to pass through vertices so our embedded nodes generally cannot be coinciding with mesh vertices (except at the endpoints). In practice, this is typically not an issue since the weights can be very small placing the embedded node almost arbitrarily close to a vertex. These restrictions can be alleviated with approaches such as [Sifakis et al. 2007a; Wang et al. 2014]. This makes our approach better than global remeshing algorithms since we will never add smaller triangles even in presence of cuts that are very close, whereas any global remeshing algorithm will require to have smaller triangles in this region when it tries to align the mesh with all the cuts.

Our method is a straightforward albeit useful combination of the virtual node algorithm and constraint stabilization. Our examples model non-dynamic creases. However, since the virtual node algorithm can be used to cut a mesh dynamically as shown in [Molino et al. 2004] and our constraint matrix can be trivially recomputed based on the new cut mesh, the extension of our method to dynamically added creases should be straightforward. The cost of dynamically adding a crease would be the cost of running the virtual node algorithm (or any other cutting algorithm) on the crease along with the cost of recomputing the W matrix, and both of these steps are quite efficient.

Acknowledgements

Research was supported in part by ONR N00014-13-1-0346, ONR N00014-11-1-0707, ARL AHPCRC W911NF-07-0027, and the Intel Science and Technology Center for Visual Computing. Computing resources were provided in part by ONR N00014-05-1-0479. S.P. was supported by the Stanford Graduate Fellowship.

References

BANDO, Y., KURATATE, T., AND NISHITA, T. 2002. A simple method for modeling wrinkles on human skin. In *Proc. of the 10th Pacific Conf. on Comput. Graph. and Applications*.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *ACM SIGGRAPH 98*, ACM Press/ACM SIGGRAPH.

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. Tracks: Toward directable thin shells. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07.

BICKEL, B., BOTSCH, M., ANGST, R., MATUSIK, W., OTADUY, M., PFISTER, H., AND GROSS, M. 2007. Multi-scale capture of facial geometry and motion. SIGGRAPH '07.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 28–36.

BURGOON, R., WOOD, Z. J., AND GRINSPUN, E. 2006. Discrete shells origami. In *21st International Conference on Computers and Their Applications, CATA-2006, Seattle, Washington, USA, March 23-25, 2006, Proceedings*.

CERDA, E., AND MAHADEVAN, L. 2003. Geometry and physics of wrinkling. *Phys. Rev. Lett. 90* (Feb).

CUTLER, L. D., GERSHBEIN, R., WANG, X. C., CURTIS, C., MAIGRET, E., PRASSO, L., AND FARSON, P. 2007. An art-directed wrinkle system for cg character clothing and skin. *Graph. Models 69*.

FLYNN, C., AND MCCORMACK, B. A. O. 2008. Finite element modelling of forearm skin wrinkling. *Skin Research and Tech.*.

GOULEKAS, K. E. 2001. *Visual Effects in a Digital World: A Comprehensive Glossary of over 7,000 Visual Effects Terms*, 1st ed. Morgan Kaufmann Publishers Inc.

HOGBEN, L., Ed. 2007. *Handbook of Linear Algebra*, 1st ed. Chapman and Hall/CRC Press, Boca Raton.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*

JIMENEZ, J., ECHEVARRIA, J. I., OAT, C., AND GUTIERREZ, D. 2011. *GPU Pro 2*. AK Peters Ltd., ch. Practical and Realistic Facial Wrinkles Animation.

KAUFMANN, P., MARTIN, S., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2009. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph.*.

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*.

KILIAN, M., FLÖRY, S., CHEN, Z., MITRA, N. J., SHEFFER, A., AND POTTMANN, H. 2008. Curved folding. *ACM Trans. Graph.*.

KIM, D., KOH, W., NARAIN, R., FATAHALIAN, K., TREUILLE, A., AND O'BRIEN, J. F. 2013. Near-exhaustive precomputation of secondary cloth effects. *ACM Trans. Graph. 32*, 87:1–87:8.

LI, P., AND KRY, P. G. 2014. Multi-layer skin simulation with adaptive constraints. In *Proceedings of the Seventh International Conference on Motion in Games*, 171–176.

MAGNENAT-THALMANN, N., KALRA, P., LUC LEVEQUE, J., BAZIN, R., BATISSE, D., AND QUERLEUX, B. 2002. A computational skin model: Fold and wrinkle formation. *Trans. Info. Tech. Biomed. 6*.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 385–392.

MÜLLER, M., AND CHENTANEZ, N. 2010. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10.

NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics*, 147:1–10.

NARAIN, R., PFAFF, T., AND O'BRIEN, J. F. 2013. Folding and crumpling adaptive sheets. *ACM Trans. Graph. 32*, 4, 51:1–51:8.

O'BRIEN, J., AND HODGINS, J. 1999. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH 99*, vol. 18.

PARKE, F. I., AND WATERS, K. 2008. *Computer Facial Animation, second edition*. AK Peters, Ltd.

RÉMILLARD, O., AND KRY, P. G. 2013. Embedded thin shells for wrinkle simulation. *ACM Trans. Graph. 32*, 4, 50:1–50:8.

ROHMER, D., POPA, T., CANI, M.-P., HAHMANN, S., AND SHEFFER, A. 2010. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. SIGGRAPH ASIA '10.

SELLE, A., LENTINE, M., AND FEDKIW, R. 2008. A mass spring model for hair simulation. *ACM Transactions on Graphics*.

SELLE, A., SU, J., IRVING, G., AND FEDKIW, R. 2009. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Trans. on Vis. and Comput. Graph.*.

SHINAR, T., SCHROEDER, C., AND FEDKIW, R. 2008. Two-way coupling of rigid and deformable bodies. SCA '08, 95–103.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. (SIGGRAPH Proc.)*.

SIFAKIS, E., DER, K., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 81–90.

SOLOMON, J., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2012. Flexible developable surfaces. *Comp. Graph. Forum*.

TERAN, J., BLEMKER, S., NG, V., AND FEDKIW, R. 2003. Finite volume methods for the simulation of skeletal muscle. In *Proc. of 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. *Proc. of 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*.

TERZOPOULOS, D., AND WATERS, K. 1990. Physically-based facial modeling, analysis, and animation. *J. Vis. and Comput. Anim. 1* (december), 73–80.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Proc. SIGGRAPH 87*.

WANG, H., HECHT, F., RAMAMOORTHI, R., AND O'BRIEN, J. F. 2010. Example-based wrinkle synthesis for clothing animation. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10.

WANG, Y., JIANG, C., SCHROEDER, C., AND TERAN, J. 2014. An Adaptive Virtual Node Algorithm with Robust Mesh Cutting. In *Eurographics/ACM SIGGRAPH Symp. on Comp. Anim.*

WEINSTEIN, R., TERAN, J., AND FEDKIW, R. 2006. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE TVCG 12*, 3, 365–374.

ZHANG, Y., AND SIM, T. 2005. Realistic and efficient wrinkle simulation using an anatomy-based face model with adaptive refinement. In *Proc. of the Comp. Graph. International 2005*.