Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2009)
E. Grinspun and J. Hodgins (Editors)

A Point-based Method for Animating Elastoplastic Solids

Dan Gerszewski   Haimasree Bhattacharya   Adam W. Bargteil

University of Utah

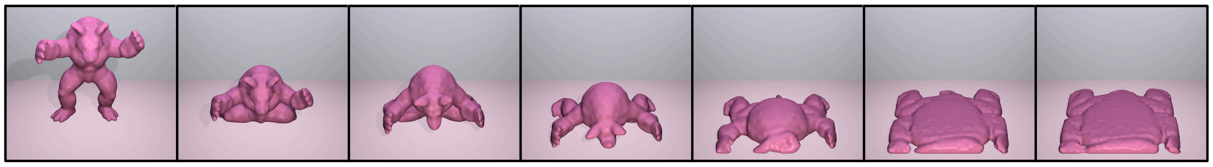

Figure 1: *An armadillo demonstrates non-Newtonian behavior similar to a cornstarch solution—resisting large stresses, it initially bounces on the ground, but when the stress is reduced it flows readily.*

Abstract

In this paper we describe a point-based approach for animating elastoplastic materials. Our primary contribution is a simple method for computing the deformation gradient for each particle in the simulation. The deformation gradient is computed for each particle by finding the affine transformation that best approximates the motion of neighboring particles over a single timestep. These transformations are then composed to compute the total deformation gradient that describes the deformation around a particle over the course of the simulation. Given the deformation gradient we can apply arbitrary constitutive models and compute the resulting elastic forces. Our method has two primary advantages: we do not store or compare to an initial rest configuration and we work directly with the deformation gradient. The first advantage avoids poor numerical conditioning and the second naturally leads to a multiplicative model of deformation appropriate for finite deformations. We demonstrate our approach on a number of examples that exhibit a wide range of material behaviors.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation.
[point-based simulation, natural phenomena, physics-based animation.]

1. Introduction

Materials that incorporate both plastic and elastic deformations such as chewing gum, toothpaste, shaving cream, sauces, bread dough, and modeling clay are frequently encountered in everyday life and have recently been successfully used in special effects such as the honey in *Bee Movie* [Rui07] and the food in *Ratatouille* [GRPS07]. In fact the later work won the Visual Effects Society award for *Outstanding Effects in an Animated Motion Picture*. At the same time, point-based simulation methods have increased dramatically in popularity and sophistication in recent years. These methods are capable of modeling a wide range of materials in a variety of contexts, from real-time fluid simulations [BMF07] to fracturing solids [PKA*05]. Their versatility makes them especially attractive for computer graphics applications.

In this paper we describe a point-based approach for animating elastoplastic materials. Our primary contribution is a simple method for computing the deformation gradient for each particle in the simulation. The deformation gradient is computed for each particle by finding the affine transformation that best approximates the motion of neighboring particles over a single timestep. This transformation is found using a least-squares fit to the positions of neighboring particles at the beginning and end of the timestep. These transformations are then composed to compute the total deforma-

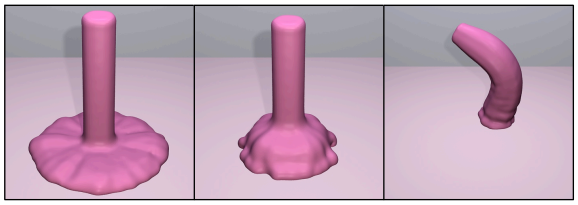

Figure 2: *Three cylinders with different material properties fall on the ground.*

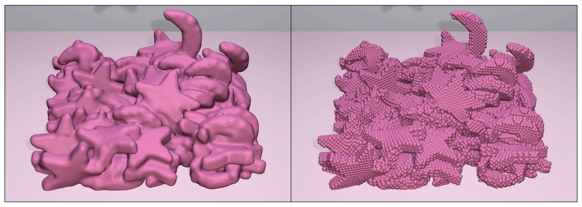

Figure 3: *Two different shapes form a pile on the ground. The right image shows the simulation particles.*

tion gradient that describes the deformation around a particle over the course of the simulation.

Our approach has two primary advantages. First, we do not store and compare to an initial rest state. Under large plastic deformations the mapping from an initial rest state to the current state becomes numerical ill-conditioned. By storing only the elastic part of the deformation we avoid these numerical problems. Second, instead of working with a strain metric, we work directly with the deformation gradient. By focusing on the deformation gradient our approach can handle arbitrary constitutive models [ITF04]. More importantly, working with the deformation gradient naturally leads to a multiplicative formulation of deformation, which is more suitable to finite deformations than the additive models from classical plasticity that are often used in graphics [SH98]. We demonstrate our approach on a number of examples that exhibit a wide range of material behaviors.

2. Related Work

A complete survey of point-based methods is beyond the scope of this paper, but we heartily recommend the book by Gross and Pfister [GP07]. We focus our attention on methods for animating elastoplastic solids and viscoelastic fluids. Terzopoulos and Fleisher [TF88] introduced inelastic deformations, including viscoelasticity, plasticity, and fracture to the graphics community. O'Brien and colleagues [OBH02] incorporated a similar plasticity model into a finite element simulation to animate ductile fracture. To avoid problems with poorly conditioned or tangled elements they demonstrated only limited amounts of plastic deformation. We also note that they used an additive model of plasticity. While such a model is appropriate when considering infinitesimal deformations, as Irving and colleagues [ITF04] pointed out, a multiplicative model is more appropriate in the context of finite plastic deformation. Clavet et al. [CBP05] modeled viscoelastic fluids with a mass-spring system in which the springs are dynamically inserted and removed. Their springs explicitly model viscous and elastic forces and include a model of plastic flow. Goktekin et al. [GBO04] took an alternative approach and added elastic forces to an Eulerian fluid simulation. In their approach, a linear strain rate is integrated through time and undergoes plastic decay. Their approach used a linear model of elastic deformation that is not invariant to rotations. This shortcoming was addressed by Losasso and colleagues [LSSF06] who applied a rotation to the advected elastic strain to account for rotations in the velocity field. However, as noted by Irving [Irv07], because this model is not based on the deformation gradient it is unable to model hyperelastic materials.

Müller et al. [MKN*04] introduced a point-based method for animating elastic, plastic, and melting objects. They broke the possible deformations into two separate regimes. Deformations that were largely elastic were treated by comparing the current configuration of neighboring particles to a rest configuration, while large plastic deformations were handled by updating a strain measure in a way similar to Goktekin et al. [GBO04]. Their approach to primarily elastic deformation uses moving least-squares to fit a transformation that maps neighbors in a reference shape to the current shape. While we use a very similar moving least-squares fit to compute the deformation gradient, we fit the deformation over individual timesteps and compose the deformations to arrive at the total deformation. While this approach invariably leads to drift in the deformation gradient over time, it is able to handle changing neighborhoods and large plastic flow in a unified way. Keiser et al. [KAG*05] also developed a unified approach by substituting fluid dynamics for the large plastic deformation regime of Müller et al. [MKN*04]. Like ours, their approach is able to model a wide variety of materials in a unified way.

Recently, Solenthaler and colleagues [SSP07] have replaced the moving least-squares approach used by Müller et al. [MKN*04] and Keiser et al. [KAG*05] with an SPH formulation. Their approach is able to model fluids, elastic, and rigid objects as well as objects that have parts of different types. Additionally, they include melting and solidification, merging and splitting, and plasticity using the model of O'Brien et al. [OBH02]. More recently, Hieber and Koumoutsakos [HK08] described a Lagrangian particle method for simulating linear and nonlinear elastic solids that does not require a rest configuration. Instead of performing a least-squares fit to the deformation in every timestep, they update the deformation gradient by integrating the gradient of the velocity field. In contrast to these meshless methods, Bargteil et al. [BWHT07] introduced a finite element

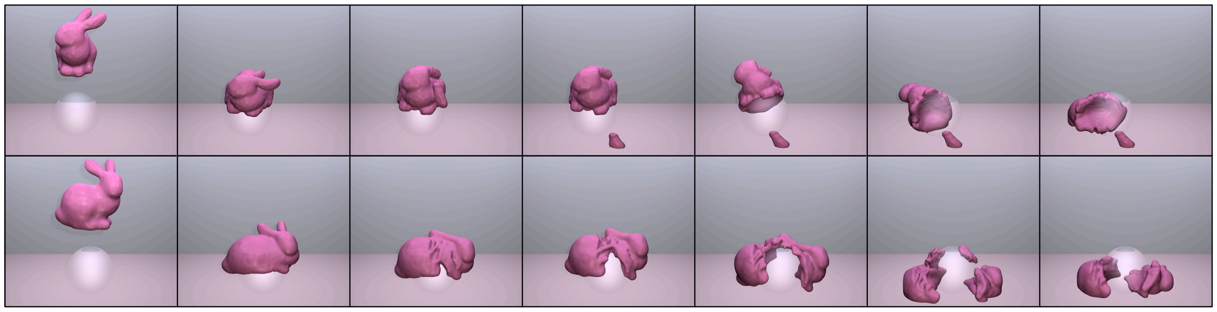
Figure 4: *An elastoplastic bunny falls on a sphere. An additional example appears in the video.*

method for animating large viscoplastic flow. Their approach relied on a robust remeshing operation to maintain well-conditioned elements. Wojtan and Turk [WT08] improved on this approach by using embedded surface meshes, producing highly detailed animations of heavily deformed objects. By using embedded meshes they were also able to adopt a fast and simple remeshing procedure. These last two papers are the only work in graphics that shares both the main advantages of our approach. However, our approach has the advantage of being meshless, allowing us to avoid remeshing and the consequent resampling and smoothing of simulation variables.

3. Method

In this section we describe our method for computing the deformation gradient and the consequent elastic forces. We focus on the modifications we made to the open-source SPH simulator released by Adams et al. [APKG07]. For additional details on SPH simulation we refer the reader to that paper and its references.

Our goal is to compute elastic forces in a point-based simulation. In order to do so, we must first compute the deformation in the vicinity of each particle, p_i. We first consider how to compute the deformation around p_i over a single timestep. Let $\mathbf{x}_i$ be the position of p_i at the beginning of the timestep and $\mathbf{y}_i$ be the location of p_i at the end of the timestep. If p_j are the neighbors of p_i, we seek the transformation matrix $\mathbf{F}$, such that

$$\sum_{j=1}^{n} \left\| \mathbf{F}(\mathbf{x}_j - \mathbf{x}_i) - (\mathbf{y}_j - \mathbf{y}_i) \right\| \tag{1}$$

is minimized. If we let

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_n \end{pmatrix} \tag{2}$$

and

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \cdots & \mathbf{y}_n \end{pmatrix}, \tag{3}$$

where the individual $\mathbf{x}_i$ and $\mathbf{y}_i$ are column vectors, then if the deformation can be represented with an affine transformation, we have

$$\mathbf{FX} = \mathbf{Y}. \tag{4}$$

Taking the transpose of both sides and multiplying both sides by $\mathbf{X}$ we obtain the normal equations,

$$\mathbf{XX}^T\mathbf{F}^T = \mathbf{XY}^T. \tag{5}$$

Solving for $\mathbf{F}$, we have

$$\mathbf{F} = \left(\left(\mathbf{XX}^T \right)^{-1} \mathbf{XY}^T \right)^T. \tag{6}$$

This solution gives us the best linear transformation for the neighborhood around p_i in a least-squares sense. However, we want nearer particles to have greater influence, so we multiply the columns of $\mathbf{X}$ and $\mathbf{Y}$ by a weighting kernel. Our implementation uses the *poly6* kernel (the default smoothing kernel given by Müller et al. [MCG03]),

$$W_{\text{poly6}}(\mathbf{r},h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - r^2\right)^3 & 0 \le r \le h \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

This approach gives us the deformation over a single timestep. However, this deformation is a linear transformation and transformations compose through multiplication. Thus, to compute the deformation over some time interval, we break the interval into a series of k timesteps, estimate the deformation over each timestep and compute

$$\mathbf{F} = \prod_{i=k}^{1} \mathbf{F}_i. \tag{8}$$

We note that only a single $\mathbf{F}$, representing the total elastic deformation, need be stored for each particle. The individual $\mathbf{F}_i$ are computed during the associated timestep, but not stored.

Once we have the deformation gradient we can apply any constitutive model we like, compute strain, stress, and elastic forces and move the simulator forward. In our implementation we diagonalize $\mathbf{F}$ into $\mathbf{U\hat{F}V^T}$ using a singular value decomposition [ITF04] and apply the multiplicative plasticity

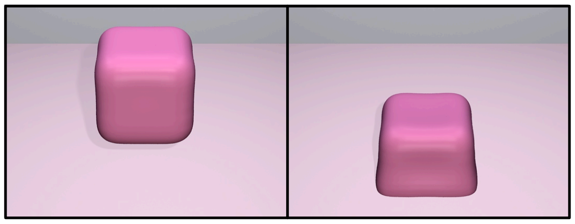

Figure 5: *Hyperelastic boxes dropped on the ground. The left cube is quite stiff, the right cube is softer.*

model described by Bargteil et al. [BWHT07] to obtain the elastic deformation, $\hat{\mathbf{F}}_e$. We then compute the diagonalized stress as in Irving et al. [ITF04],

$$\hat{\mathbf{P}} = 2\mu_e(\hat{\mathbf{F}}_e - \mathbf{I}) + \lambda \mathrm{Tr}\left(\hat{\mathbf{F}}_e - \mathbf{I}\right)\mathbf{I}. \tag{9}$$

Following Solenthaler et al. [SSP07] and accounting for our diagonalized deformation gradient and stress, the elastic force p_i exerts on p_j is

$$\mathbf{f}_{ij} = -2v_i v_j \mathbf{U}\hat{\mathbf{F}}_e\hat{\mathbf{P}}\mathbf{V}^T\mathbf{d}_{ij} \tag{10}$$

where v_i and v_j are the volumes of particles p_i and p_j and

$$\mathbf{d}_{ij} = \nabla W(\mathbf{F}_e^{-1}(\mathbf{y}_j - \mathbf{y}_i), h). \tag{11}$$

Note that the vector from y_i to y_j is back projected to the reference space before applying the weighting kernel. We use the weighting kernel developed specifically for elastic forces by Solenthaler et al. [SSP07],

$$W(\mathbf{r},h) = \begin{cases} c\frac{2h}{\pi}\cos\left(\frac{(r+h)\pi}{2h}\right) + c\frac{2h}{\pi} & 0 \le r \le h \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where

$$c = \frac{\pi}{8h^4\left(\frac{\pi}{3} - \frac{8}{\pi} + \frac{16}{\pi^2}\right)}. \tag{13}$$

In order to ensure conservation of momentum, $\mathbf{f}_{ij}$ and $\mathbf{f}_{ji}$ are averaged and equal and opposite forces are applied to the particles. We note that because we have diagonalized $\mathbf{F}$ and $\mathbf{P}$, the forces computed in Equation (10) are rotationally invariant.

4. Results & Discussion

Our implementation adds the elastic forces described in this paper to the open source SPH simulator by Adams and colleagues [APKG07]. The resulting system may be thought of as a “unified SPH” simulator and is capable of simulating liquids and solids as well as materials that demonstrate properties of both liquids and solids. In fact many of our examples included SPH pressure forces as well as elastic forces, as we found that pressure forces provided additional stability. We refer the interested reader to the paper by Adams and colleagues [APKG07] and the associated source code for details such as time integration (symplectic forward Euler), neighborhood selection (the 30 nearest neighbors within a given radius), etc.

Figure	Δt (ms)	Particles	Sec/Frame
Fig. 1	0.5	52316	96.4888
Fig. 4	0.1	40556	379.065
Fig. 2 (left)	0.1	16770	138.257
Fig. 2 (center)	0.1	16770	137.909
Fig. 2 (right)	0.1	16770	141.591
Fig. 5 (left)	0.1	665	4.76452
Fig. 5 (right)	0.1	665	4.88767
Fig. 7 (top, left)	0.1	12152	102.545
Fig. 7 (center, left)	0.1	12152	86.9948
Fig. 7 (bottom, left)	0.1	12152	94.4097
Fig. 7 (top, right)	0.1	12152	96.135
Fig. 7 (center, right)	0.1	12152	87.1509
Fig. 7 (bottom, right)	0.1	12152	94.3632

Table 1: *Timing results for the examples in this paper.*

Figures 1-7 demonstrate our method’s ability to handle a wide range of materials. Figure 1 shows an example with a modified version of our plasticity model that divides the flow rate by the magnitude of the stress, so that the material flows more easily under small stresses. Figure 5 demonstrates a hyper-elastic material where the eigenvalues of $\mathbf{F}$ are squared before computing the stress. Figure 6 compares one of our simulations with real-world footage of bread dough and Figure 7 demonstrates the effects of varying our material parameters. Figure 8 shows a comparison of our approach with an approach that stores and compares to a reference shape and then removes plastic deformation before computing elastic forces and an additive approach that computes Green’s strain at every timestep and adds it to the total elastic strain. As is expected, storing the reference configuration works very well for largely elastic bodies, but under large plastic flow the simulation becomes unstable. Conversely, an additive model of elastic deformation works well enough when most of the deformation is plastic, but fails to return to the rest shape when the deformation is primarily elastic.

Table 1 summarizes our computation times. All results were obtained on a single core of a Xeon E5410 (2.33 Ghz), with 16 GB of memory available. Profiling has shown that in the example in Figure 4, 14% of the computation time was spent in our elasticity code. Half of this time was spent performing eigendecompositions. This total cost is roughly twice the cost of surface tension forces, which we did not use in our examples. We note that our examples were run with very conservative timesteps—some of our examples ran successfully with 10x larger timesteps.

Generating visually appealing, time-coherent surfaces for particle-based simulations remains a difficult problem

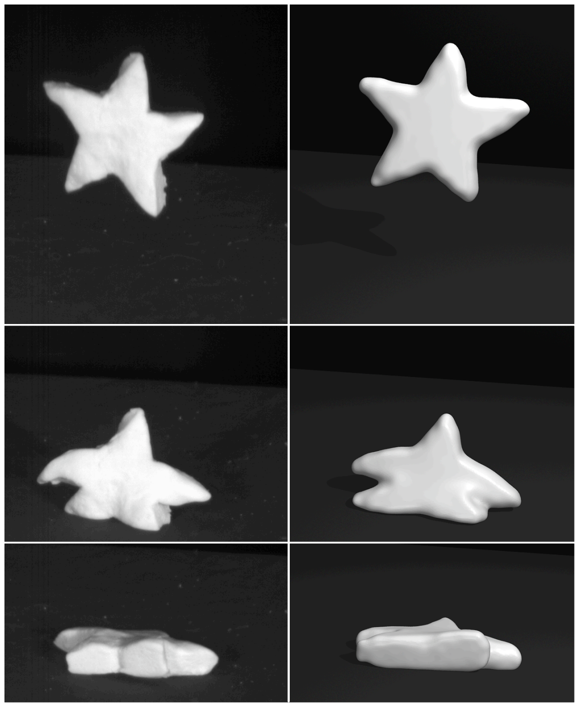

Figure 6: *Real-world footage of bread dough shaped like a star (left) is compared to a simulation (right).*

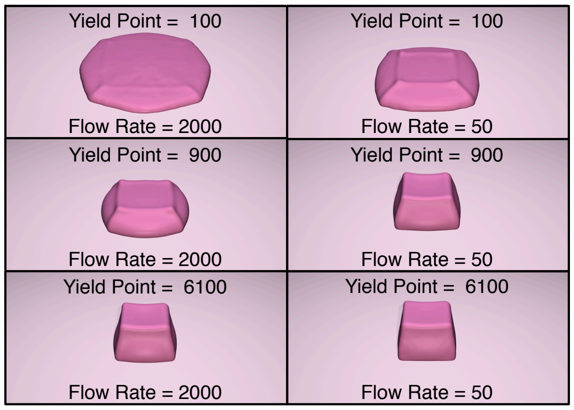

Figure 7: *We demonstrate the effects of our plastic material parameters by dropping a box on the ground.*

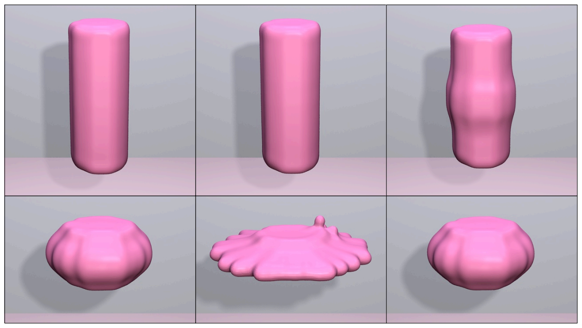

Figure 8: *Final frames in a comparison of our method (left) against a method that uses a rest configuration (middle) and a method with an additive strain model (right). The top row is an elastic material, the bottom row is a very plastic material. The simulation consists of applying and then releasing an analytic compression force that increases away from the center of the object. The lower middle image is the last frame before the simulation became unstable.*

that is beyond the scope of this paper. Our results were generated with a variation of the method described by Williams [Wil08].

One of the paramount concerns in any computer graphics simulator is stability and ours in no exception. One of the sources of error and instability in our approach is the estimation of the deformation gradient. In particular, if a particle does not have enough neighbors or the distribution of particles is degenerate, $\mathbf{X}\mathbf{X}^T$ will be ill-conditioned. To address this problem, we do not update the deformation gradient if a particle has less than a set number of neighbors (6 in our implementation), if $\mathbf{X}\mathbf{X}^T$ is ill-conditioned, or if the update would cause any of the eigenvalues of $\mathbf{F}$ to be less than or equal to zero. We also note that plastic flow tends to improve stability by bringing $\mathbf{F}$ towards the identity. Consequently, relaxing the constraint that plastic deformation be volume preserving in cases when $\mathbf{F}$ encodes large volume changes further improves stability. When the method does fail, it tends to be in areas around sharp features, where a particle's neighbors subtend a small solid angle, or in areas where topological changes are occurring. Addressing these issues is an important area of future work.

Other interesting areas of future work include implementing an implicit integrator, addressing topological changes in a physically based manner (currently, topological changes occur when particle neighborhoods change), and methods for resampling/adaptive sampling. The last direction is particularly interesting as it may improve stability as well as provide performance benefits. Additionally, while we have demonstrated our approach with a particle-based method, the general approach to computing the deformation gradient should be applicable in other simulation methods, such as Eulerian grid-based or finite element techniques.

Our approach is well-suited to simulating materials that experience large plastic deformations. It is also capable of simulating rather stiff elastic materials, though some drift is inevitable. Unfortunately, our approach is not well-suited to the large elastic deformations exhibited by soft objects. In such cases the deformation gradient becomes ill-conditioned and our method breaks down. For simulating such materials the invertible finite element approach developed by Irving et al. [ITF04] is more appropriate.

We believe our approach has a number of advantages over competing techniques. In particular, it does not require any rest configuration, no remeshing is needed, it can handle elastic and large plastic deformations in a unified framework and it is simple to implement and inexpensive to compute.

Acknowledgments

We would like to thank Bart Adams for releasing his SPH software and Chris Wojtan and Jessica Hodgins for capturing the bread dough footage in Figure 6. We would also like to thank the anonymous reviewers and the Utah graphics group for their helpful comments.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (2007), 48.

[BMF07] BRIDSON R., MÜLLER-FISCHER M.: Fluid simulation: Siggraph 2007 course notes. In *ACM SIGGRAPH 2007 courses* (2007), pp. 1–81.

[BWHT07] BARGTEIL A. W., WOJTAN C., HODGINS J. K., TURK G.: A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. 26*, 3 (2007), 16.

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *The Proccedings of the Symposium on Computer Animation* (2005), pp. 219–228.

[GBO04] GOKTEKIN T. G., BARGTEIL A. W., O'BRIEN J. F.: A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 3 (2004), 463–468.

[GP07] GROSS M., PFISTER H.: *Point-Based Graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[GRPS07] GOKTEKIN T. G., REISCH J., PEACHEY D., SHAH A.: An effects recipe for rolling a dough, cracking an egg and pouring a sauce. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches* (New York, NY, USA, 2007), ACM, p. 67.

[HK08] HIEBER S. E., KOUMOUTSAKOS P.: A Lagrangian particle method for the simulation of linear and nonlinear elastic models of soft tissue. *J. Comp. Phys. 227*, 21 (2008), 9195–9215.

[Irv07] IRVING G.: *Methods for the Physically Based Simulation of Solids and Fluids*. PhD thesis, Stanford University, 2007.

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *The Proceedings of the Symposium on Computer Animation* (2004), pp. 131–140.

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRÉ P., GROSS M.: A unified Lagrangian approach to solid-fluid animation. In *The Proceedings of the Symposium on Point-Based Graphics* (2005), pp. 125–133.

[LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. *ACM Trans. Graph. 25*, 3 (2006), 812–819.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *The Proceedings of the Symposium on Computer Animation* (2003), pp. 154–159.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *The Proceedings of the Symposium on Computer Animation* (2004), pp. 141–151.

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21*, 3 (2002), 291–294.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ; P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (2005), 957–964.

[Rui07] RUILOVA A.: Creating realistic cg honey. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters* (New York, NY, USA, 2007), ACM, p. 58.

[SH98] SIMO J., HUGHES T.: *Computational Inelasticity*. Springer-Verlag, 1998.

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Journal of Visualization and Computer Animation 18*, 1 (2007), 69–82.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *The Proceedings of ACM SIGGRAPH* (1988), pp. 269–278.

[Wil08] WILLIAMS B.: *Fluid Surface Reconstruction from Particles*. Master's thesis, University of British Columbia, 2008.

[WT08] WOJTAN C., TURK G.: Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27*, 3 (2008), 1–8.