Ductile Fracture for Clustered Shape Matching

Ben Jones*
University of Denver

April Martin
University of Utah

Joshua A. Levine
Clemson University

Tamar Shinar
University of California, Riverside

Adam W. Bargteil†
Univeristy of Maryland, Baltimore County

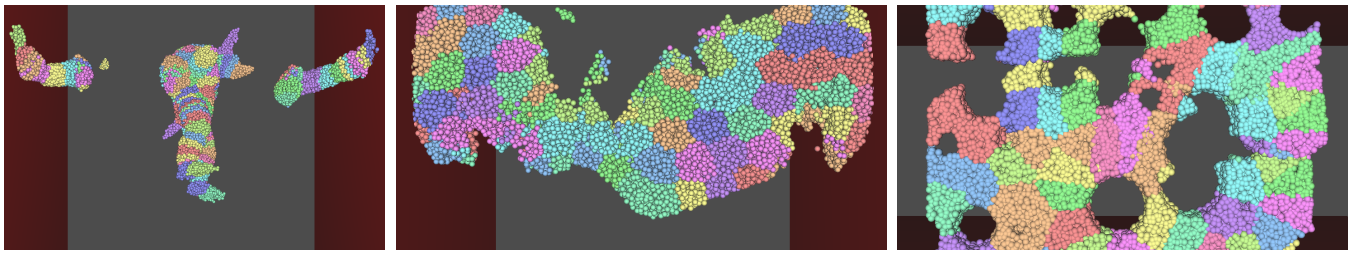

Figure 1: *Left: An armadillo is tortured by being torn apart by the arms while being fired upon by spherical projectiles. Center:A heart-breaking example. Right: Tearing a slice of swiss cheese.*

Abstract

In this paper, we incorporate ductile fracture into the clustered shape matching simulation framework for deformable bodies, thus filling a gap in the shape matching literature. Our plasticity and fracture models are inspired by the finite element literature on deformable bodies, but are adapted to the clustered shape matching framework. The resulting approach is fast, versatile, and simple to implement.

Keywords: Shape Matching, Ductile Fracture

Concepts: •**Computing methodologies** → **Simulation by animation; Physical simulation;**

1 Introduction

Shape matching is a geometrically motivated technique for animating deformable bodies introduced a decade ago by Müller and colleagues [2005]. Figure 2 summarizes the approach. In their seminal work, Müller and colleagues [2005] introduced a simple plasticity model and, in follow up work, Rivers and James [2007] incorporated a simple fracture model. However, *ductile fracture*, the combination of plastic deformation and fracture, has not yet been addressed in the shape matching literature. Moreover we adopt more general and sophisticated approaches to both plasticity and fracture.

In this paper, we enable the animation of ductile fracture by incorporating plasticity and fracture models into the clustered shape matching framework. Our models are inspired by finite element approaches for animating deformable bodies, but are adapted to clustered shape matching. Specifically, inspired by the work of Irving and colleagues [2004] and Bargteil and colleagues [2007], we introduce a multiplicative plasticity model that incorporates yield stress, flow rate, and work hardening. Inspired by the work of O'Brien and colleagues [1999; 2002], we introduce a cluster-based fracture approach that splits individual clusters along the plane orthogonal to the direction the cluster is most stretched. Taken together these contributions allow animation of ductile fracture in the clustered shape matching framework, as demonstrated in Figure 1.

*email: benjones@cs.du.edu
†email: adamb@umbc.edu

I3D '16,, March 01 2016, TBA,
ISBN: 978-1-4503-4043-4/16/03
DOI: http://dx.doi.org/10.1145/2856400.2856415...$15.00

2 Related Work

The geometrically motivated shape matching approach was introduced by Müller and colleagues [2005], who demonstrated impressive results and described the key advantages of the approach: efficiency, stability, and controllability. Given these advantages, shape matching is especially appealing in interactive animation contexts such as video games. The authors also introduced several extensions including linear and quadratic deformations (in addition to rigid deformations), cluster-based deformation, and plasticity.

Two years later, Rivers and James [2007] introduced lattice-based shape matching, which used a set of hierarchical lattices to define the shape matching clusters. They took advantage of the regular structure of the lattices to achieve extremely high performance. They also incorporated a simple fracture model that removed over-extended links in the lattice. More recently, Bargteil and Jones [2014] incorporated strain-limiting into clustered shape matching. In follow-up work, Jones and colleagues [2015] explored improved clustering strategies and introduced simple collision proxies for the clusters. We incorporate our ductile fracture approach into their open-source framework and use their approaches for strain limiting, clustering, and collision detection.

In order to improve stability for non-volumetric objects (i.e. shells and strands) and to simplify sampling, Müller and Chentanez extended shape matching to track particle orientation [Müller and Chentanez 2011b]. They also introduced a simple model for plastic deformation. This approach proved to be useful for animating clothing and hair attached to animated characters [Müller and Chentanez 2011a].

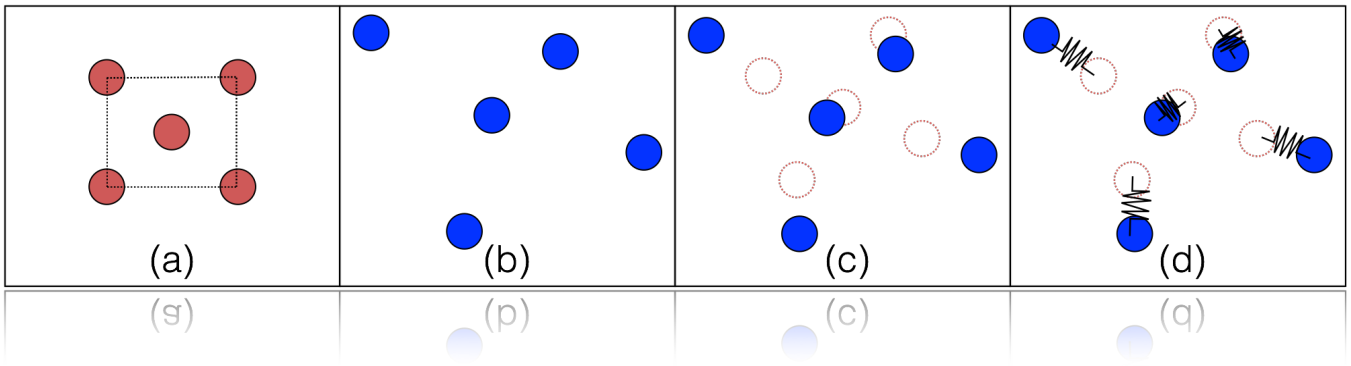

Figure 2: *Shape Matching Overview: (a) An object (here, a square) is sampled with particles, p_i, to get rest positions, $\mathbf{r}_i$. (b) As particles are subjected to external forces and constraints, their positions, $\mathbf{x}_i$, are updated in world space. (c) The best-fitting rigid transformation of the particles' rest positions, $\mathbf{r}_i$, to their world positions, $\mathbf{x}_i$ is computed. The dotted red circles are the* goal *positions, $\mathbf{g}_i$. (d) Hookean springs pull the world positions toward the goal positions.*

Other popular approaches to real-time animation of deformable solids include position-based dynamics [Müller et al. 2007; Bender et al. 2014; Macklin et al. 2014; Kelager et al. 2010], frame-based models [Gilles et al. 2011; Faure et al. 2011], and projective dynamics [Liu et al. 2013; Bouaziz et al. 2014]. While plasticity and fracture have been demonstrated for position-based dynamics, such phenomena invalidate the precomputations that make frame-based models and projective dynamics computationally efficient. As argued by Bargteil and Jones [2014], the shape matching framework has some advantages over position-based dynamics; in particular better adherence to Newton's first and second laws of motion.

Ductile fracture is distinguished from brittle fracture by the inclusion of plastic deformation. Materials undergoing ductile fracture (e.g. play dough) appear to *tear*, while brittle materials (e.g. glass) appear to *shatter*. Most real-world materials demonstrate some amount of plastic deformation during failure, so purely brittle models have fairly limited application in computer animation. Both plasticity and fracture were first demonstrated in computer animation by the pioneering work of Terzopoulos and Fleischer [1988]; however, it was O'Brien and colleagues [2002] who first combined these phenomena to animate ductile fracture. Since that time, plasticity and fracture have remained very active research areas in computer animation and a thorough review is beyond the scope of this short paper.

Our approach to fracture closely resembles that of O'Brien and colleagues [1999; 2002]; however, instead of splitting tetrahedra, we split shape matching clusters. Our plasticity model closely resembles that of Bargteil and colleagues [2007], except that we apply it to shape matching clusters instead of individual tetrahedra and make no particular effort to ensure that plastic deformation does not lead to instability, i.e. we do not update clusters to ensure well-conditioned matrices as done by, for example, Jones and colleagues [2014]. While this can be done in the proposed framework, in practice we have found it sufficient to limit the total plastic deformation through work hardening and other plasticity parameter adjustment.

3 Methods

For completeness and readability, we first briefly review the shape matching approach of Müller and colleagues [2005] before introducing our plasticity and fracture models. Finally we briefly discuss our approaches to sampling and clustering.

3.1 Shape Matching

In the shape matching framework objects are discretized into a set of particles, $p_i \in \mathcal{P}$, with masses, m_i, and rest positions, $\mathbf{r}_i$, that follow a path, $\mathbf{x}_i(t)$, in world-space through time. Shape matching takes its name from the fact that, each frame, we match the rest shape to the deformed shape by finding the least-squares best-fit rigid transformation from the rest pose to the current deformed pose by solving for the rotation matrix, $\mathbf{R}$, and translation vector, $\bar{\mathbf{x}} - \bar{\mathbf{r}}$, that minimizes

$$\sum_i m_i \|\mathbf{R}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) - \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\|^2. \tag{1}$$

The best translation is given by the center-of-mass in the rest ($\bar{\mathbf{r}}$) and world ($\bar{\mathbf{x}}$) space. Computing the rotation, $\mathbf{R}$, is more involved. We first compute the least-squares best-fit linear deformation gradient, $\mathbf{F}$. Specifically, we seek the $\mathbf{F}$ that minimizes

$$\sum_i m_i \|\mathbf{F}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) - \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\|^2. \tag{2}$$

Setting the derivative with respect to $\mathbf{F}$ to 0 and re-arranging terms we arrive at

$$\mathbf{F} = \left(\sum_i m_i \mathbf{O}(\mathbf{x}_i, \mathbf{r}_i)\right)\left(\sum_i m_i \mathbf{O}(\mathbf{r}_i, \mathbf{r}_i)\right)^{-1} = \mathbf{A}_{xr}\mathbf{A}_{rr}^{-1}, \tag{3}$$

where $\mathbf{O}(\cdot,\cdot)$ is the outer product matrix

$$\mathbf{O}(\mathbf{a}_i, \mathbf{b}_i) = \left(\mathbf{a}_i - \bar{\mathbf{a}}\right)\left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^T, \tag{4}$$

and $\mathbf{A}_{**}$ is a convenient shorthand. We then compute $\mathbf{R}$ using the polar decomposition,

$$\mathbf{F} = \mathbf{RS} = \left(\mathbf{U}\mathbf{V}^T\right)\left(\mathbf{V}\mathbf{\Sigma}\mathbf{V}^T\right) \tag{5}$$

where $\mathbf{S} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T$ is a symmetric matrix and $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the singular value decomposition (SVD) of $\mathbf{F}$. While several researchers (e.g. [Rivers and James 2007]) have pointed out that polar decompositions can be computed faster than the SVD, especially when warm started, we use the SVD for its robustness and for our plasticity and fracture models (see Sections 3.2 and 3.3). We also note that we diverge slightly from Müller and colleagues [2005], following instead Jones and colleagues [2015], and compute the polar

decomposition of $\mathbf{F}$, not the left matrix ($\mathbf{A}_{xr}$). This modification is particularly important if the distribution of mass in the cluster is non-uniform and $\mathbf{F}$ is not a pure rotation.

Given $\mathbf{R}$ and $\bar{\mathbf{x}} - \bar{\mathbf{r}}$, we define goal positions, $\mathbf{g}_i$, as

$$\mathbf{g}_i = \mathbf{R}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) + \bar{\mathbf{x}}. \tag{6}$$

Hookean springs are then used to define forces that move the particles toward the goal positions.

Clustered Shape Matching Breaking an object into multiple overlapping clusters allows for richer and more localized deformations. Fortunately, handling multiple clusters is straightforward. When computing a particle's contribution to cluster quantities, we divide the particle's mass among the clusters to which it belongs. For a particle p_i in cluster $c \in \mathcal{C}$ we introduce a weight w_{ic} that describes how much of p_i's mass is contributed to cluster c and replace m_i with $w_{ic}m_i$ in equations (1)-(3) and when computing cluster mass and center-of-mass. Specifically, if particle p_i belongs to n_i clusters, then the center-of-mass of cluster c, $\bar{\mathbf{x}}_c$, is

$$\bar{\mathbf{x}}_c = \frac{\sum_{p_i \in \mathcal{P}_c}(w_{ic}m_i)\,\mathbf{x_i}}{\sum_{p_i \in \mathcal{P}_c}(w_{ic}m_i)}, \tag{7}$$

where $\mathcal{P}_c$ is the set of particles in cluster c. Furthermore, when computing the goal position, $\mathbf{g}_i$, for a particle we perform a weighted average of the goal positions given by each cluster to which it belongs. That is,

$$\mathbf{g}_i = \sum_c w_{ic}\,\mathbf{g}_{ic}, \tag{8}$$

where $\mathbf{g}_{ic}$ is the goal position for particle p_i in cluster c.

Clustering We use the clustering method of Jones and colleagues [2015]. This method is a variation of the fuzzy c-means algortithm and produces overlapping clusters where each particle may belong to several clusters to varying degrees. As in the popular k-means clustering algorithm, this algorithm alternates between updating cluster membership and updating cluster centers. However, updating membership involves updating weights, w_{ic}, and cluster centers are the weighted center-of-mass of members. Please see Jones and colleagues [2015] for more details including analysis of different weighting functions, varying cluster size, degree of overlap, etc.

Strain Limiting To maintain stability we adopt the strain limiting approach advocated by Bargteil and Jones [2014]. However, in the presence of plastic deformation (see Section 3.2) we typically increase the maximum allowed stretch (γ in their paper) to avoid instabilities when clusters disagree about the current rest shape.

Collision Handling We use the approach of Jones and colleagues [2015] for handling collisions. Their approach uses spheres intersected with half-spaces as collision proxies for clusters, which is very well-suited to our fracture approach that divides clusters with planes.

Sampling Geometry The distribution of particles that model an object affects the resulting simulation. We experimented with both grid-based and blue noise sampling and preferred the results from blue noise over the highly structured grid-based sampling. Our blue noise sampler is based on Bridson's fast Poisson disk sampling [Bridson 2007]. In both cases, for objects whose boundary is an arbitrary manifold, we simply sample particles within the bounding box of the object and discard particles outside the surface.

3.2 Plasticity

Our approach to plastic deformation adapts the model of Bargteil and colleagues [2007] to the clustered shape matching framework. To accommodate plastic deformation we store and update an additional matrix, $\mathbf{F}_c^p$, for each cluster, c. For readability we drop the subscript, but the following is computed for each cluster. We then compute the elastic part of the deformation gradient

$$\mathbf{F}^e = \mathbf{F}\left(\mathbf{F}^p\right)^{-1}, \tag{9}$$

where $\mathbf{F}$ is given by Equation (3). We then decompose $\mathbf{F}^e$ in Equation (5).

$\mathbf{F}^p$ is initialized to the identity, $\mathbf{I}$. Then each timestep we compute the volume preserving part of the diagonalized $\mathbf{F}^e$,

$$\mathbf{F}^* = \det(\mathbf{\Sigma}^e)^{-1/3}\mathbf{\Sigma}^e. \tag{10}$$

We then compare

$$\|\mathbf{F}^* - \mathbf{I}\|_F \tag{11}$$

to a plastic yield threshold, λ, where $\|\cdot\|_F$ is the Frobenius norm. If the threshold is not exceeded $\mathbf{F}^p$ remains unchanged. Otherwise we update $\mathbf{F}^p$ by

$$\mathbf{F}^p_{new} = \left(\mathbf{F}^*\right)^\gamma \mathbf{V}\mathbf{F}^p_{old}, \tag{12}$$

where $\mathbf{V}$ is the matrix of right singular vectors in Equation (5) and γ is given by

$$\gamma = \min\left(\frac{\nu * \|\mathbf{F}^* - \mathbf{I}\|_F - \lambda - K\alpha}{\|\mathbf{F}^* - \mathbf{I}\|_F}, 1\right), \tag{13}$$

where ν and K are user-given flow rate and work hardening/softening constant, respectively, and α is a measure of cumulative stress that is initialized to zero and then updated by

$$\dot{\alpha} = \|\mathbf{F}^e - \mathbf{I}\|_F. \tag{14}$$

We do not apply additional left-hand rotations when computing $\mathbf{F}^p_{new}$ as these would be discarded during the decomposition in Equation (5).

3.3 Fracture

To model fracture, we allow shape matching clusters to split into two clusters. Each timestep, for each cluster, we compare the largest singular value, $\Sigma_{\max}$, computed in Equation (5), to a toughness, τ_c, which can vary between clusters. If $\Sigma_{\max} > \tau_c$ then the cluster is added to a priority queue with priority $\Sigma_{\max} - \tau_c$. We also record the corresponding right singular vector $\mathbf{V}_{\max}$. The fracture process occurs at the end of the timestep, after computing dynamics. We iteratively remove clusters from the priority queue and, if the $\Sigma_{\max}$ still exceeds τ_c, we split the cluster into two clusters as follows.

We assume that the fracture surface is defined by the plane that passes through the center of mass of the cluster, $\bar{\mathbf{x}}_c$, and is orthogonal to the singular vector $\mathbf{V}_{\max}$. Particles on the positive side of this plane are removed from the cluster and added to a new cluster.

To propagate fractures to nearby clusters, we examine each particle in the split cluster. If a particle is a member of clusters on both sides of the fracture plane, it is split into two particles. The mass is divided between the two new particles based on how many clusters they belong to; all other properties are copied to both new particles.

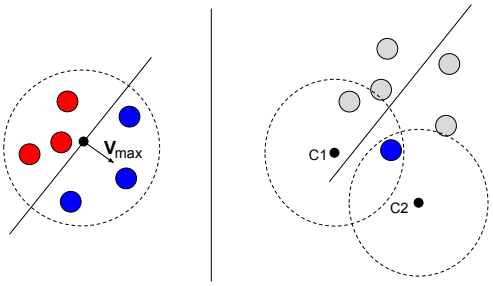

Figure 3: *Left: a cluster is fractured along a splitting plane defined by its center of mass and the vector $\mathbf{V}_{max}$. Right: splits are propagated to nearby clusters. Since the blue particle is a member of clusters on both sides of the fracture plane, it is split into two new particles, one in cluster $c1$, one in cluster $c2$.*

This process mimics duplication of node degrees of freedom in finite element approaches to fracture. To compare which side of the plane the particle and clusters are on, we evaluate

$$\text{sign}\left((\mathbf{x}_i - \bar{\mathbf{x}}_c) \cdot \mathbf{V}_{\max}\right) \neq \text{sign}\left((\bar{\mathbf{x}}_d - \bar{\mathbf{x}}_c) \cdot \mathbf{V}_{\max}\right). \quad (15)$$

for each cluster, d, of which particle i is a member. This process is illustrated in Figure 3. We also split "outlier" particles, which, even after strain limiting, are still too far from a cluster center. In this case, we create a new particle associated only with the cluster and dissociate the cluster from the original particle.

Initially our clusters are well sampled; however, fractures may create degenerate clusters. To prevent instability due to these clusters, we compute $\mathbf{A}_{rr}^{-1}$ through the pseudoinverse. If, when computing the pseudoinverse, any of the singular values are thresholded, we set $\mathbf{F}_c^p = \mathbf{I}$.

Because cluster membership may have changed due to the fracturing of a different cluster, we verify that $\Sigma_{\max} > \tau_c$ before fracturing and also recompute $\mathbf{V}_{\max}$. A remaining issue is that after fracture it may take several timesteps for elastic forces to reduce the strain in fractured clusters. Consequently, at the next timestep $\Sigma_{\max}$ may still be greater than τ_c resulting in additional fracture. This process tends to continue, leading to clusters breaking up into individual particles in a few timesteps—an artifact we refer to as "particle spray." This artifact is less problematic for animations of brittle fracture, but is undesirable in ductile fracture. We have developed two approaches to addressing the artifact. The simplest is to disallow fracture for a cluster that has fractured until $\Sigma_{\max} < \tau_c$. The second, more flexible, approach boosts τ_c after a fracture. Specifically,

$$\tau_c = \tau_0(1 + \tau_b)e^{\tau_f t}, \quad (16)$$

where τ_0 is the initial toughness for the cluster, τ_b is the maximal boost, and τ_f controls how quickly the boost falls off. Both approaches effectively address the artifact.

As clusters fracture and particles split, some clusters may represent negligible mass or may have too few particles to meaningfully contribute to the simulation. Thus, when a clusters mass goes below a threshold or it has less then four member particles we delete it from the system. Due to cluster removal, some particles may not belong to any cluster; these are also removed from the system.

4 Results and Discussion

We have used our method to animate a number of examples of ductile fracture. Timing results taken on a Macbook Pro with a 2.4GHz Intel i5 processor are given in table Table 1. In all particle renderings, particle color is used to indicate the nearest cluster center. We stress that clusters overlap and that particles typically belong to more than one cluster. Furthermore, though particles decrease in mass when they split during fracture, we do not modify the size of particles in our renderings. Consequently, fractured clusters may appear to have much larger mass than they actually do. We also note that, to better demonstrate our technique many of our examples are simplified and do not include gravity; this choice was made consciously and does not reflect any limitation of our technique.

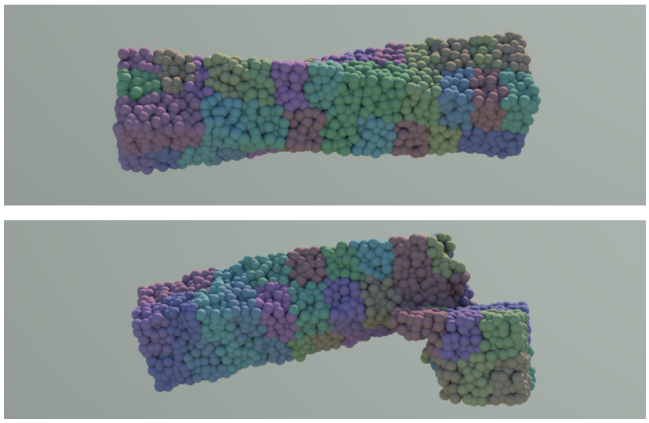

Figure 4: *A bar is twisted and then the clamps are released. Top: A tough plastic bar does not fracture. Bottom: The bar fractures before the clamps are released.*

In our first example, an armadillo is tortured by simultaneously pulling on and firing cannonballs at its arms (see Figure 1). Our second example is more didactic; a simple bar is twisted (see Figure 4). First we demonstrate plastic deformation, which allows the bar to maintain its twisted shape, then we enable fracture. This twisting deformation causes the two large tears in the material, one which separates the object into two pieces, and a large partial tear. In our third example a projectile is fired through a thin sheet of material with various material properties (see Figure 5). The weak, brittle material breaks into many small pieces. The strong, brittle material breaks into larger pieces. The ductile material deforms plastically near the impact before fracturing, bending permanently in the wake of the projectile. In our heartbreaking fourth example a complex shape is pulled apart (see Figure 1). The complex shape determines where the tear begins and how it propagates. Similarly, in our final example the holes in a slice of swiss cheese determine how the model is torn (see Figure 1). In our final example, a hollow ball is dropped on the ground and fractures (see Figure 6).

Limitations and Future Work Our approach has a number of limitations that provide ample potential for future work in clustered shape matching. While our examples typically run at or near real-time, one limitation of our current implementation is that the computational cost is somewhat variable; many fractures occurring at once can have a very significant impact on performance. In the future it would be interesting to explore asynchronous fracturing where, instead of emptying the priority queue every timestep, it is processed as computational resources allow. Moreover, the clustered shape matching framework is particularly amenable to parallelization, but we have not yet explored this topic.

Like many fracture methods in computer animation, we do not have a good solution to generating fractured geometry. Like previous shape matching work [Müller et al. 2005; Rivers and James 2007],

Table 1: *Timing results in ms per frame taken on a Macbook Pro with a 2.4Ghz Intel i5 processor.*

example	# particles	dynamics	plasticity	fracture	total
armadillo	20115	16	< 1	< 1	24
twisted bar	5317	7	< 1	0	7
twisted bar with fracture	5317	7	< 1	< 1	9
projectile through ductile plate	5325	20	< 1	3	27
broken heart	20132	22	< 1	<1	31
swiss cheese	25032	27	< 1	<1	39

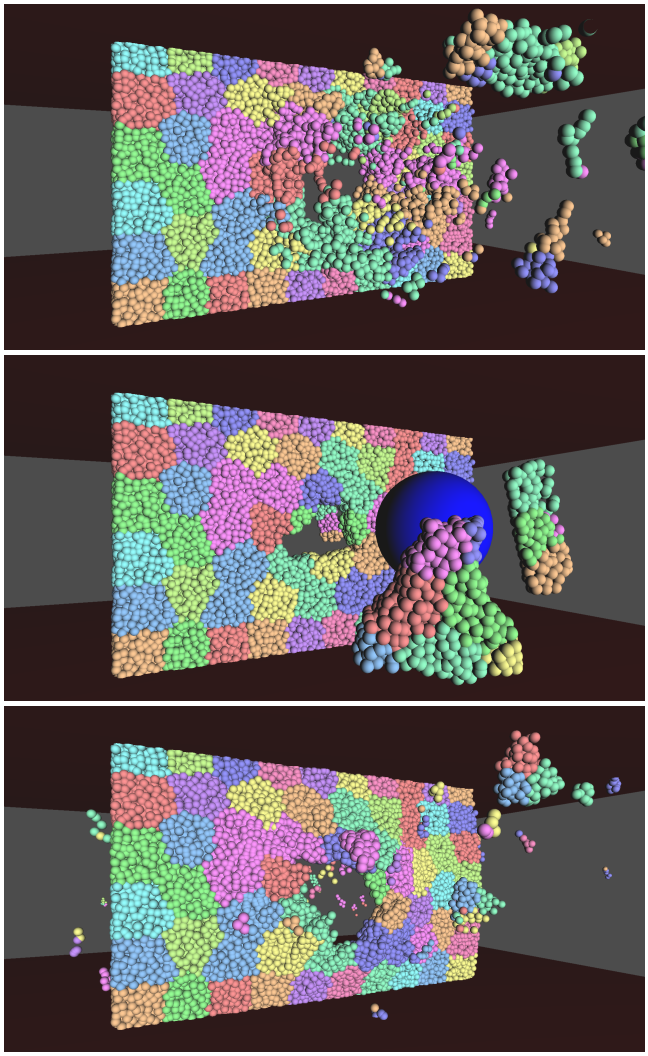

Figure 5: *A projectile is fired through a thin sheet of material with varying material properties. top: weak and brittle; center: strong and brittle; bottom: strong and ductile. Note permanent the plastic deformation near the hole.*

we can embed a high-resolution render mesh for unbreakable objects, but creating new geometry when fracture occurs is more difficult. Currently we use an off-the-shelf particle skinning tool [Bhattacharya et al. 2015] to skin the simulation particles. This approach suffers two major drawbacks: the results are low-resolution and the fracture surfaces are overly smooth. One approach to generating fracture geometry is to simply render the simulation geometry and increase the resolution where fracture occurs [O'Brien and Hodgins 1999; O'Brien et al. 2002]. The simulation geometry can can

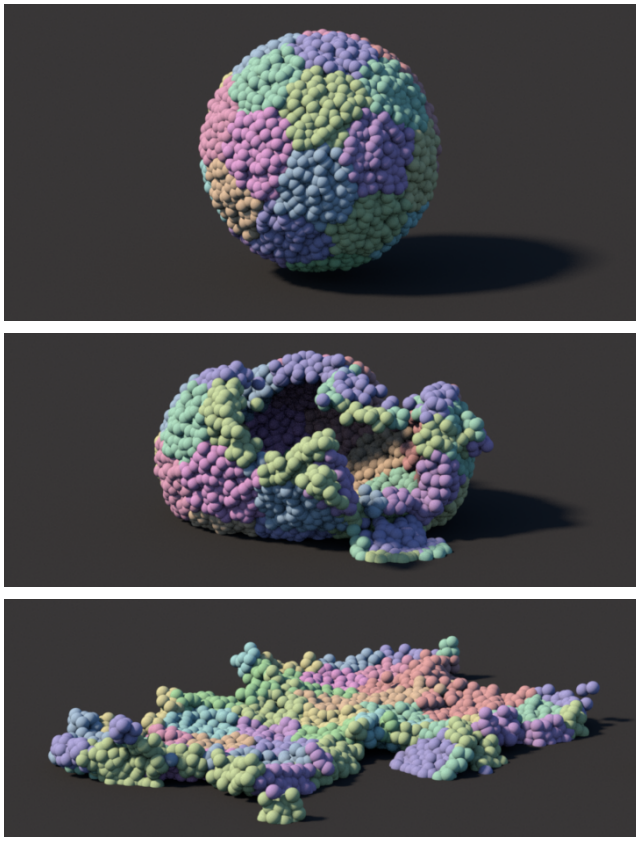

Figure 6: *Three frames from a sequence where a hollow ball is dropped on the ground and fractures.*

later be coarsened to improve computational efficiency [Pfaff et al. 2014]. This approach leads to somewhat unpredictable costs and is, at present, too slow for the interactive applications where shape matching excels. Levine and colleagues [2014] suggested a number of other techniques for generating fracture geometry for spring-mass systems that may be applicable in our interactive context. Exploring these ideas is an interesting avenue for future work.

Our blue-noise sampling improves upon the regular grids and is effective for our purposes, but better approaches certainly exist. In particular, it would be interesting to explore adaptive sampling so that computational resources can be focused on interesting areas of the object. Changing the sampling over time as done by Pauly and colleagues [2005] is also a promising avenue for future work, which may help address the geometric limitations discussed above.

The biggest limitation of our approach is a lack of theoretical underpinnings for the clustered shape matching framework; we do not yet have any mathematical tools to analyze the approach. We

do not really understand how the method behaves as particle counts or timesteps decrease or as the cluster size or number of clusters change. This limitation does not mean the approach is not useful. After all, the finite element method was in use for decades before a mathematical framework was developed to analyze its properties. In a similar way, we believe the clustered shape-matching framework will prove extremely useful in practice while researchers develop mathematical tools for analysis.

Conclusion One of the primary advantages of the clustered shape matching approach is that the number of degrees of freedom is much larger then the number of "integration units"—clusters in this case. The opposite is true of finite element methods with unstructured meshes where the number of tetrahedra is often considerably larger than the number of vertices. For graphical applications, visual detail, which correlates with the number of degrees of freedom, is of paramount importance and computation, which correlates with "integration units," is often limited. For these reasons, the clustered shape matching framework is extremely appealing for computer animation, especially interactive animation. The utility and versatility of this framework is greatly improved by our extensions to the animation of ductile fracture.

Acknowledgements

The authors wish to thank the anonymous reviewers for their time and helpful comments. This work was supported in part by National Science Foundation awards IIS-1314896, IIS-1314757, and IIS-1314813

References

BARGTEIL, A. W., AND JONES, B. 2014. Strain limiting for clustered shape matching. In *Proceedings of the Seventh International Conference on Motion in Games*, ACM, New York, NY, USA, MIG '14, 177–179.

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

BENDER, J., MÜLLER, M., OTADUY, M. A., TESCHNER, M., AND MACKLIN, M. 2014. A survey on position-based simulation methods in computer graphics. *Computer Graphics Forum*, 1–25.

BHATTACHARYA, H., GAO, Y., AND BARGTEIL, A. W. 2015. A level-set method for skinning animated particle data. *IEEE Trans. Vis. Comput. Graph. 21* (Mar), 315–327.

BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July), 154:1–154:11.

BRIDSON, R. 2007. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, ACM, New York, NY, USA, SIGGRAPH '07.

FAURE, F., GILLES, B., BOUSQUET, G., AND PAI, D. K. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30*, 4 (July), 73:1–73:10.

GILLES, B., BOUSQUET, G., FAURE, F., AND PAI, D. K. 2011. Frame-based elastic models. *ACM Trans. Graph. 30*, 2 (Apr.), 15:1–15:12.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 131–140.

JONES, B., WARD, S., JALLEPALLI, A., PERENIA, J., AND BARGTEIL, A. W. 2014. Deformation embedding for point-based elastoplastic simulation. *ACM Trans. Graph. 33*, 2 (Apr.), 21:1–21:9.

JONES, B., MARTIN, A., LEVINE, J. A., SHINAR, T., AND BARGTEIL, A. W. 2015. Clustering and collision detection for clustered shape matching. In *Proceedings of ACM Motion in Games*, ACM, New York, NY, USA.

KELAGER, M., NIEBE, S., AND ERLEBEN, K. 2010. A Triangle Bending Constraint Model for Position-Based Dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation*, K. Erleben, J. Bender, and M. Teschner, Eds.

LEVINE, J. A., BARGTEIL, A. W., CORSI, C., TESSENDORF, J., AND GEIST, R. 2014. A peridynamic perspective on spring-mass fracture. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

LIU, T., BARGTEIL, A. W., O'BRIEN, J. F., AND KAVAN, L. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics 32*, 6 (Nov.), 209:1–7. Proceedings of ACM SIGGRAPH Asia 2013, Hong Kong.

MACKLIN, M., MÜLLER, M., CHENTANEZ, N., AND KIM, T.-Y. 2014. Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4, 104.

MÜLLER, M., AND CHENTANEZ, N. 2011. Adding physics to animated characters with oriented particles. In *Workshop in Virtual Reality Interactions and Physical Simulation*, The Eurographics Association, 83–91.

MÜLLER, M., AND CHENTANEZ, N. 2011. Solid simulation with oriented particles. In *ACM transactions on graphics (TOG)*, vol. 30, ACM, 92.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (July), 471–478.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2, 109–118.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *The Proceedings of ACM SIGGRAPH 99*, 137–146.

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

PAULY, M., KEISER, R., ADAMS, B., DUTRÉ;, P., GROSS, M., AND GUIBAS, L. J. 2005. Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3, 957–964.

PFAFF, T., NARAIN, R., DE JOYA, J. M., AND O'BRIEN, J. F. 2014. Adaptive tearing and cracking of thin sheets. *ACM Transactions on Graphics 33*, 4 (July), xx:1–9. To be presented at SIGGRAPH 2014, Vancouver.

RIVERS, A. R., AND JAMES, D. L. 2007. Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3 (July).

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *The Proceedings of ACM SIGGRAPH*, 269–278.