Reclustering for Large Plasticity in Clustered Shape Matching

Michael Falkenstein
University of Maryland, Baltimore County

Ben Jones
University of Utah

Joshua A. Levine
University of Arizona

Tamar Shinar
University of California, Riverside

Adam W. Bargteil
University of Maryland, Baltimore County

ABSTRACT

In this paper, we revisit the problem online reclustering in clustered shape matching simulations and propose an approach that employs two nonlinear optimizations to create new clusters. The first optimization finds the embedding of particles and clusters into three-dimensional space that minimizes elastic energy. The second finds the optimal location for the new cluster, working in this *embedded space*. The result is an approach that is more robust in the presence of elastic deformation. We also experimentally verify that our clustered shape matching approach converges as the number of clusters increases, suggesting that our reclustering approach does not change the underlying material properties. Further, we demonstrate that particle resampling is not strictly necessary in our framework allowing us to trivially conserve volume. Finally, we highlight an error in estimating rotations in the original shape-matching work [Müller et al. 2005] that has been repeated in much of the follow up work.

CCS CONCEPTS

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

KEYWORDS

Shape Matching, Clustering, Plasticity

ACM Reference Format:
Michael Falkenstein, Ben Jones, Joshua A. Levine, Tamar Shinar, and Adam W. Bargteil. 2017. Reclustering for Large Plasticity in Clustered Shape Matching. In *Proceedings of MiG '17*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3136457.3136473

1 INTRODUCTION

Shape matching is a geometrically motivated technique for animating deformable bodies introduced a decade ago by Müller and colleagues [2005]. Figure 1 summarizes the approach. The basic approach samples a deformable object with particles, which determine the degrees of freedom in the object. Each timestep, a best-fit rigid transformation of the rest *shape* of the object to the current configuration of particles is computed and Hookean springs are used to pull the particles toward the rigidly transformed shape. A powerful extension to this basic approach, also introduced by Müller and colleagues [2005], is to break the object into several overlapping clusters. We refer to this approach as *clustered shape matching*. Having more than one cluster imbues the object with a richer space of deformation, while overlap keeps the object from falling apart. While this approach lacks a well-developed mathematical underpinning, it has a number of advantages that make it especially well-suited to interactive graphics applications, such as video games.

Recently, Chentanez and colleagues [2016] extended basic clustered shape matching with the ability to dynamically add and remove clusters and particles. These extensions allow them to handle extremely large plastic deformations that otherwise cause previous approaches [Jones et al. 2016] to fail. In this paper, we revisit this problem and propose an alternative approach to reclustering that employs two nonlinear optimizations to create new clusters. The first optimization finds the embedding of particles and clusters into three-dimensional space that minimizes elastic energy. The second finds the optimal location for the new cluster, working in this *embedded space*. The result is an approach that is more robust in the presence of elastic deformation. We also experimentally verify that our approach converges as the number of clusters increases, suggesting that our reclustering approach does not change the underlying material properties. Further, we demonstrate that particle resampling is not strictly necessary in our framework allowing us to trivially conserve volume. Finally, we highlight an error in estimating rotations in the original shape-matching work [Müller et al. 2005] that has been repeated in much of the follow up work.

2 BACKGROUND

In this section we first review related work and then provide a brief overview of the clustered shape matching approach before detailing our reclustering approach in Section 3.

2.1 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

MiG '17, November 8–10, 2017, Barcelona, Spain

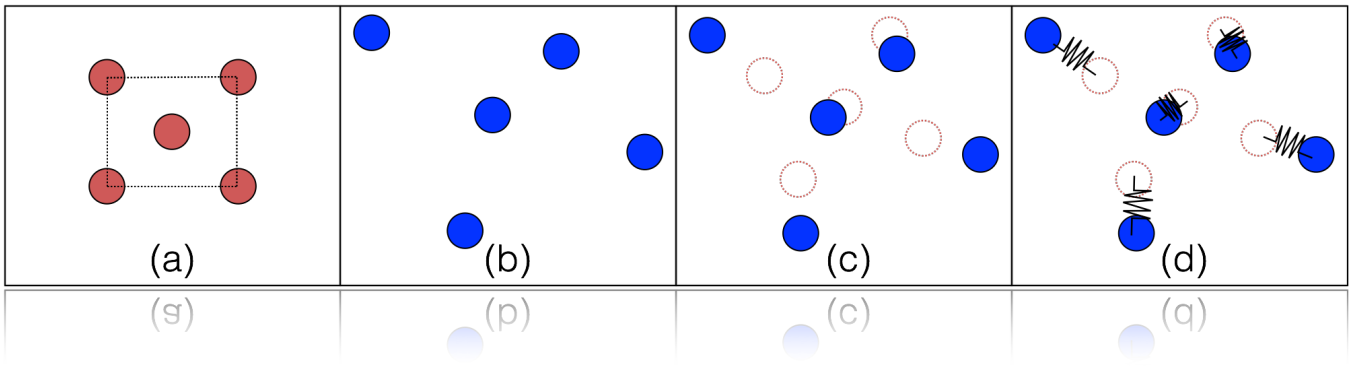

Figure 1: *Shape Matching Overview: (a) An object (here, a square) is sampled with particles, p_i, to get rest positions, $\mathbf{r}_i$. (b) As particles are subjected to external forces and constraints, their positions, $\mathbf{x}_i$, are updated in world space. (c) The best-fitting rigid transformation 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.*

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. The following year Steinemann [2008] adapted the lattice based approach to octrees enabling spatial adaptivity. 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 2011]. This approach proved to be useful for animating clothing and hair attached to animated characters [Müller and Chentanez 2011].

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. Jones and colleagues [2016] later introduced a plasticity model and approach to fracture enabling ductile fracture in the clustered shape matching framework.

Other popular approaches to real-time animation of deformable solids include position-based dynamics [Bender et al. 2014; Kelager et al. 2010; Macklin et al. 2014; Müller et al. 2007], frame-based models [Faure et al. 2011; Gilles et al. 2011], and projective dynamics [Bouaziz et al. 2014; Liu et al. 2013]. 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.

2.2 Clustered Shape Matching

For completeness and readability, we first briefly review the shape matching approach of Müller and colleagues [2005] and the extensions of Bargteil, Jones, and colleagues [2014; 2015; 2016] before introducing our plasticity and fracture models. Finally we briefly discuss our approaches to sampling and clustering.

2.3 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 model (see Section 2.3.1).

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.

2.3.1 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

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

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 algorithm 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 2.3.1) 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 an object whose boundary is an arbitrary manifold, we simply sample particles within the bounding box of the object and discard particles outside the surface.

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 $\mathbf{F}^e$ as 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 constants, 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).

We note that, in the presence of plasticity, the optimal rotation $\mathbf{R}$ should be found by taking the polar decomposition of the elastic part of the deformation gradient,

$$\mathbf{F}^e = \mathbf{A}_{xr}\mathbf{A}_{rr}^{-1}\left(\mathbf{F}^p\right)^{-1} \tag{15}$$

and that when computing goal positions we must account for the plastic deformation

$$\mathbf{g}_i = \mathbf{R}\mathbf{F}^p\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) + \bar{\mathbf{x}}. \tag{16}$$

Regrettably, these details were omitted by Jones and colleagues [2016].

3 METHODS

3.1 The Best Rotation

We begin by explicitly highlighting an error in early shape matching work. Mueller and colleagues [2005] mistakenly stated that because $\mathbf{A}_{rr}$ is symmetric it does not affect the rotation, $\mathbf{R}$, which leads them to compute the rotation from $\mathbf{A}_{xr}$, ignoring $\mathbf{A}_{rr}$. This produces the same rotation as the polar decomposition of $\mathbf{F}$ if $\mathbf{A}_{rr}$ is diagonal or $\mathbf{F}$ has a condition number of 1. However, if $\mathbf{A}_{rr}$ is not diagonal *and* $\mathbf{F}$ includes a non-uniform scale, the polar decompositions of $\mathbf{A}_{xr}$ and $\mathbf{F}$ result in different rotations.

To make this clear we consider a concrete example. For $\mathbf{A}_{rr}$ to be non-diagonal, we have to have some asymmetry of our shape with respect to the reference coordinate system. So we consider a two-dimensional rectangle aligned with the $x = y$ axis. Specifically we sample the four corners: $(1, 3)$, $(3, 1)$, $(-1, -3)$, and $(-3, -1)$. The basis matrix is then

$$\mathbf{A}_{rr}^{-1} = \begin{pmatrix} 20 & 12 \\ 12 & 20 \end{pmatrix}^{-1} = \frac{1}{64}\begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix}. \tag{17}$$

If we stretch our rectangle by a factor of 2 in the x dimension, then

$$\mathbf{A}_{xr} = \begin{pmatrix} 40 & 24 \\ 12 & 20 \end{pmatrix}, \tag{18}$$

which is not symmetric. The polar decomposition of $\mathbf{A}_{xr}$ yields

$$\mathbf{R} = \begin{pmatrix} .9806 & -.19611 \\ .19611 & .9806 \end{pmatrix}^{-1}, \tag{19}$$

when in fact the transformation contained no rotation.

$$\mathbf{F} = \mathbf{A}_{xr}\mathbf{A}_{rr}^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \tag{20}$$

recovers the transformation. This error has persisted in the shape matching literature [Choi 2014; Müller and Chentanez 2011; Rivers and James 2007; Steinemann et al. 2008]. In practice, the error is probably not very significant. After all artists generally align symmetries of their models with the coordinate axes resulting in $\mathbf{A}_{rr}$ matrices that are nearly diagonaland the object in our example is going to begin rotating as it undoes the deformation anyway. Indeed, in our experiments we were unable to produce an example where the incorrect rotation produced visually implausible results. More significantly, the assumption that the polar decomposition of $\mathbf{A}_{xr}$ yields the optimal rigid rotation is the motivation for the elaborate plasticity model developed by Choi [2014] and adopted by Chentanez and colleagues [2016]. Consequently we adopt the simpler plasticity model of Jones and colleagues [2016], which is based on the correct rotation. As noted above, in the presence of plasticity, the optimal rotation is computed from the polar decomposition of $\mathbf{F}^e$.

3.2 Reclustering

When an object undergoes plastic deformation its rest state will in general no longer be embeddable in three-dimensional space. While for modest plastic deformations this fact can be encoded by storing per-cluster plastic offsets ($\mathbf{F}^p$), under large plastic deformations it no longer makes sense to store rest positions of particles. Instead, for each cluster we store the relative position of each member particle to the cluster center, that is $\mathbf{p}_{ic}$ replaces $\mathbf{r}_i - \bar{\mathbf{r}}$ throughout. Note that while $\mathbf{p}_{ic}$ may change to account for shifting cluster membership, it does not account for plastic deformation that occurs over the lifetime of a cluster and we are still required to store $\mathbf{F}_c^p$ for each cluster c. This modification saves computation time but does require additional storage. Another view is that the rest center-of-mass of each cluster is at the origin. Chentanez and colleagues [2016] adopted the same storage.

3.2.1 Cluster Removal. We remove clusters when the condition number of their $\mathbf{F}^p$ matrix reaches a threshold indicating that the rest state of the cluster has become significantly distorted. This trigger is similar to the approach of Bargteil and colleagues [2007] who globally remeshed whenever an individual tetrahedron's condition number exceeded a threshold, but diverges from the approach of Chentanez and colleagues [2016] who used the Frobenius norm of $\mathbf{F}^p$ or particle membership statistics to trigger cluster removal. The Frobenius norm measures the absolute magnitude of $\mathbf{F}^p$, which is quite limited because, to preserve volume, $\det \mathbf{F}^p = 1$. In contrast, the condition number measures the relative squash and stretch induced on the rest space by $\mathbf{F}^p$, making it a more appropriate measure of the cluster's condition. Moreover, because we must invert $\mathbf{F}^p$, numerical problems will arise as $\mathbf{F}^p$ becomes singular. By removing clusters when the condition number of $\mathbf{F}^p$ is large, we avoid singular matrices.

Once a cluster is marked for removal it gradually fades out over a user-specified number of timesteps by gradually reducing the weights w_{ic}. The mass of particles that were members of the cluster is gradually redistributed to other clusters as the denominator in Equation (2) gradually decreases. Changing weights shifts cluster centers, which requires updating relative positions of cluster members to keep the center of mass at the origin.

3.2.2 Cluster Addition. When adding clusters, we iterate over the particles. Any particle that soon will not be a member of any cluster (i.e. all its clusters are marked for removal) is chosen as a seed location for a new cluster. We then perform a local embedding of the rest space into an *embedded space* and then optimize the position of the cluster in this embedded space. We first describe the optimization we use to compute embedded locations, then how we determine which clusters and particles are embedded, and finally the cluster optimization.

Embedding Optimization. There are a number of viable spaces in which to add new clusters. Chentanez and colleagues [2016]

add new clusters in world space, including as members any particles that fall within a specified radius. This approach is analogous to the world space remeshing performed by Bargteil and colleagues [2007]. As pointed out by Wicke and colleagues [2010] this approach is problematic if there are large elastic deformations and world space differs greatly from the minimum-stress configuration of the object. Consequently, they perform a non-linear optimization to achieve a minimum-stress configuration and perform remeshing in this configuration. Notably, they used a local remeshing algorithm that avoided the smoothing caused by the global wholesale remeshing employed by Bargteil and colleagues [2007]. More recently, Jones and colleagues [2014] proposed linearizing the optimization problem, finding the best least-squares embedding into three-dimensional space and performing nearest neighbor queries in this embedded space. In our case, we seek embedded positions $\mathbf{e}_i$ and $\mathbf{e}_c$ for particles and clusters, respectively that minimize the elastic energy. Specifically,

$$\underset{\mathbf{e}_i,\mathbf{e}_c}{\operatorname{argmin}} \sum_{i,c} w_{ic} \|\mathbf{R}\mathbf{F}_c^p \mathbf{p}_{ic} - (\mathbf{e}_i - \mathbf{e}_c)\|^2. \tag{21}$$

To optimize Equation (21) we use the strain limiting approach of Bargteil and colleagues [2014] with the maximum allowed stretch (γ) set to zero. This approach is also very similar to most position-based dynamics implementations [Müller et al. 2007]. Twenty Jacobi iterations seems to be more than sufficient to find a good embedding. We consider the optimization converged when the sum of the squared change in particle positions is 10^{-8} times that in the first iteration. We did not experiment with alternative solvers or convergence criteria.

We initially experimented with linearizing Equation (21) as Jones and colleagues [2014] did, which amounts to ignoring the rotation, $\mathbf{R}$ and results in three decoupled $n \times n$ linear systems (where n is the sum of the number of particles and the number of clusters). However, we found that this approach yielded poor embeddings—the magnitude of the sum in Equation (21) was roughly two orders of magnitude larger than our nonlinear optimization.

Local Embedding. Jones and colleagues [2014] performed a global embedding because all particles in their simulation needed to update their neighbor lists. In our case we seek to add a single cluster at a time. Consequently we compute a local embedding that is likely to contain all the particles in the system that would be inside the cluster in embedded space. Intuitively, we find the set of all the clusters that contain the candidate particle and their neighboring clusters, where "neighbors" is defined as sharing a particle, and all the particles in these clusters. Specifically, if $\mathcal{P}_c$ is the set of particles that are members of cluster c, and $\mathcal{C}_p$ is the set of clusters of which our candidate particle, p, is a member then the set of clusters to embed, $\mathcal{C}_e$, is

$$\mathcal{C}_e = \mathcal{C}_p \cup \{c \in \mathcal{C} \mid (\exists d \in \mathcal{C}_p)\,(\exists q \in c)\,[q \in d]\} \tag{22}$$

and the set of particles to embed, $\mathcal{P}_e$, is

$$\mathcal{P}_e = \{q \in \mathcal{P} \mid (\exists c \in \mathcal{C}_e)\,[q \in c]\}. \tag{23}$$

Cluster Optimization. Once we have computed embedded positions, we initialize a new cluster center at the candidate particle's position in embedded space and run the same clustering optimization (see Section 2.3.1) as during initialization holding all other cluster centers and weights fixed. In a small number of iterations the cluster center settles into a locally optimal position. It is critical that the weights reflect the eventual state of clusters being removed or added, not the current state; if a cluster has been marked for removal, its weights should be treated as zero during the optimization and if a cluster is being added, its weights should be treated as though the cluster had fully faded in.

Like Chentanez and colleagues [2016] we initialize $\mathbf{F}^p = 0$. Because our embedding optimization results in small embedding error, residual plastic deformation is low, making zero a reasonable approximation. We do not need to estimate $\mathbf{F}^e$ because the elastic deformation is already, and more accurately, defined by the mapping from embedded space to world space. As with removing clusters, we gradually increase the weight of the new cluster over several timesteps. Once a cluster is added we continue iterating through the particles looking for additional candidate clusters. A particle may be a candidate at the beginning of this loop and be added to another cluster before becoming a candidate initialization point. Particles could be periodically shuffled to avoid this bias. Also note that a particle being selected as a seed location for a cluster does not ensure the particle ends up inside the cluster. However, in our experiments, particles are added to clusters after a few timesteps. Because of this fact, removed clusters fade out over more timesteps than added clusters fade in.

4 RESULTS

We begin with two didactic examples to demonstrate the robustness of our approach. In these examples a compression force is applied to a beam for 4 seconds and released. Specifically at position (x, y, z) the force is a $f(x, y, z) = s \cdot (-x, 0.0, 0.0)$ for some scale factor, s. In our first example the plastic yield threshold, λ, is set to zero, so that all volume preserving deformation is plastic. In the video results we can observe that without reclustering the simulation becomes unstable. In the second example we increased the scale factor, s, and set $\lambda = 0.1$ to allow for some elastic deformation. In the accompanying video we compare our approach to an approach that reclusters in world space and selects new cluster centers from among the particles that are not in the minimum number of clusters. This second approach is similar to Chentanez and colleagues [2016], but omits the critical estimate of $\mathbf{F}^e$ from neighboring clusters. The final frames from these examples using our approach are shown in figure Figure 2. Without optimizing the embedding space or the cluster location the simulation is more prone to spurious oscillations. Quantitative results using our unoptimized research code are given in Table 1. Compared to no reclustering the highly plastic example our reclustering algorithm roughly doubles the cost of the simulation. This cost could probably be reduced by tuning optimization convergence thresholds, which we left at conservative values. Our approach is faster when there is less plastic deformation and, thus, less need for reclustering, when we increased the plastic yield threshold, the cost of reclustering was reduced by more than a factor of 2. It is also worth noting that our embedding optimization is

Table 1: *Timing results in ms per frame taken on a Macbook Pro with a 2.5 Ghz Intel i7 processor. The beam is discretized with 5317 particles and 200 initial clusters.*

Example	Dynamics	Plasticity	Reclustering	Embedding Optimization	Total
Compressed Beam (no reclustering)	2.07	0.07	0	0	16.17
Compressed Beam (our algorithm)	0.56	0.07	18.60	6.64	35.54
Compressed Beam (no optimization)	0.57	0.07	9.85	0	24.99
Compressed Beam (more elastic, our algorithm)	0.59	0.04	8.03	2.60	20.62
Twisted Beam	1.12	0.16	12.56	3.31	46.54

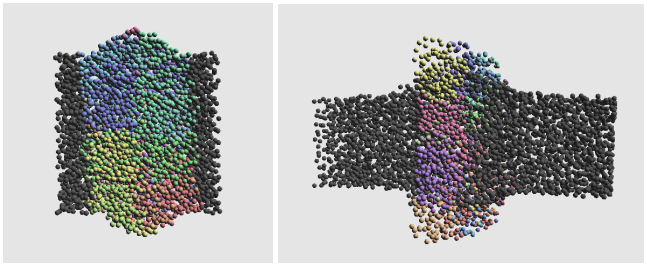

Figure 2: *A beam is compressed. Left: A highly plastic beam. Right: More elastic deformation. Colors indicate the closest cluster center. Grey particles are in their original clusters.*

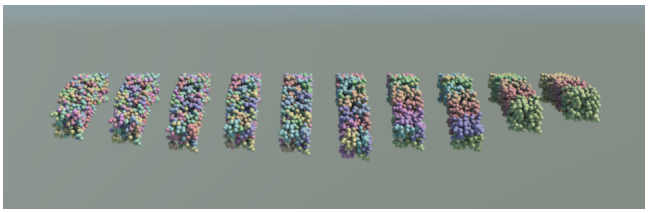

Figure 3: *Decreasing number of clusters from left to right. As the number of clusters decreases the apparent stiffness increases. The leftmost bars are visually almost indistinguishable.*

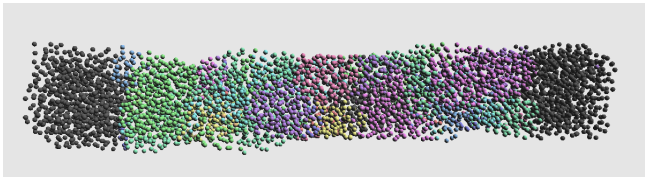

Figure 4: *A twisted plastic beam. Colors indicate the closest cluster center. Grey particles are in their original clusters.*

roughly a third of the reclustering cost. Optimizing cluster centers incurs negligible computational cost.

In Figure 3 we demonstrate the effect of increasing the number of clusters. As the number of clusters increases the simulation "converges" in the sense that it approaches some behavior. Our final example is of a twisted plastic beam.

Limitations and Future Work. An limitation of our approach when compared to Chentanez and colleagues [2016] is that we do not add or remove particles. While this choice trivially preserves volume it does not allow us to adaptively sample our objects, focusing more computational resources on visually interesting areas of the scene. We also do not have a surface tracking module and render our particles directly. We could employ the method of Bhattacharya and colleagues [2011; 2015] to skin the particles. We could also use moving least square to embed a surface mesh, though in this case the mesh may tangle under very large plastic deformations. Incorporating fracture [Jones et al. 2016] is another interesting avenue for future work.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under Grant No. IIS-1314896, IIS-1654221, and IIS-1314813.

REFERENCES

[Bargteil and Jones 2014] Adam W. Bargteil and Ben Jones. 2014. Strain Limiting for Clustered Shape Matching. In *Proceedings of the Seventh International Conference on Motion in Games*. 177–179.

[Bargteil et al. 2007] Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. 2007. A Finite Element Method for Animating Large Viscoplastic Flow. *ACM Trans. Graph.* 26, 3 (2007), 16.

[Bender et al. 2014] Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. 2014. A Survey on Position-Based Simulation Methods in Computer Graphics. *Computer Graphics Forum* (2014), 1–25.

[Bhattacharya et al. 2011] Haimasree Bhattacharya, Yue Gao, and Adam W. Bargteil. 2011. A Level-set Method for Skinning Animated Particle Data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

[Bhattacharya et al. 2015] Haimasree Bhattacharya, Yue Gao, and Adam W. Bargteil. 2015. A Level-set Method for Skinning Animated Particle Data. *IEEE Trans. Vis. Comput. Graph.* 21 (Mar 2015), 315–327. Issue 3.

[Bouaziz et al. 2014] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages.

[Bridson 2007] Robert Bridson. 2007. Fast Poisson Disk Sampling in Arbitrary Dimensions. In *ACM SIGGRAPH 2007 Sketches*. Article 22.

[Chentanez et al. 2016] Nuttapong Chentanez, Matthias Müller, and Miles Macklin. 2016. Real-time Simulation of Large Elastoplastic Deformation with Shape Matching. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 159–167.

[Choi 2014] Min Gyu Choi. 2014. Real-time simulation of ductile fracture with oriented particles. *Computer Animation and Virtual Worlds* 25, 3-4 (2014), 455–463.

[Faure et al. 2011] François Faure, Benjamin Gilles, Guillaume Bousquet, and Dinesh K. Pai. 2011. Sparse Meshless Models of Complex Deformable Solids. *ACM Trans. Graph.* 30, 4, Article 73 (July 2011), 10 pages.

[Gilles et al. 2011] Benjamin Gilles, Guillaume Bousquet, Francois Faure, and Dinesh K. Pai. 2011. Frame-based Elastic Models. *ACM Trans. Graph.* 30, 2, Article 15 (April 2011), 12 pages.

[Jones et al. 2015] Ben Jones, April Martin, Joshua A. Levine, Tamar Shinar, and Adam W. Bargteil. 2015. Clustering and Collision Detection for Clustered Shape Matching. In *Proceedings of ACM Motion in Games*.

[Jones et al. 2016] Ben Jones, April Martin, Joshua A. Levine, Tamar Shinar, and Adam W. Bargteil. 2016. Ductile Fracture for Clustered Shape Matching. In *Proceedings of the ACM SIGGRAPH symposium on Interactive 3D graphics and games*.

[Jones et al. 2014] Ben Jones, Stephen Ward, Ashok Jallepalli, Joseph Perenia, and Adam W. Bargteil. 2014. Deformation Embedding for Point-based Elastoplastic Simulation. *ACM Trans. Graph.* 33, 2, Article 21 (April 2014), 9 pages.

[Kelager et al. 2010] Micky Kelager, Sarah Niebe, and Kenny Erleben. 2010. A Triangle Bending Constraint Model for Position-Based Dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation*.

[Liu et al. 2013] Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-Spring Systems. *ACM Transactions on Graphics* 32, 6 (Nov. 2013), 209:1–7.

[Macklin et al. 2014] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified Particle Physics for Real-Time Applications. *ACM Trans. Graph.* 33, 4 (2014).

[Müller and Chentanez 2011] Matthias Müller and Nuttapong Chentanez. 2011. Adding Physics to Animated Characters with Oriented Particles. In *Workshop in Virtual Reality Interactions and Physical Simulation*. 83–91.

[Müller and Chentanez 2011] Matthias Müller and Nuttapong Chentanez. 2011. Solid Simulation with Oriented Particles. *ACM Trans. Graph.* 30, 4, Article 92 (2011), 10 pages.

[Müller et al. 2007] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *J. Vis. Comun. Image Represent.* 18, 2 (2007), 109–118.

[Müller et al. 2005] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph.* 24, 3 (July 2005), 471–478.

[Rivers and James 2007] Alec R. Rivers and Doug L. James. 2007. FastLSM: Fast Lattice Shape Matching for Robust Real-time Deformation. *ACM Trans. Graph.* 26, 3, Article 82 (July 2007).

[Steinemann et al. 2008] Denis Steinemann, Miguel A. Otaduy, and Markus Gross. 2008. Fast Adaptive Shape Matching Deformations. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 87–94.

[Wicke et al. 2010] Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. 2010. Dynamic Local Remeshing for Elastoplastic Simulation. *ACM Transactions on Graphics* 29, 4 (July 2010), 49:1–11.