APPROVAL SHEET

Title of Thesis: Reclustering for Large Plasticity in Clustered Shape Matching

Name of Candidate: Michael Falkenstein
Computer Science, 2017

Thesis and Abstract Approved: ________________________
Adam W. Bargteil
Assistant Professor
Department of Computer Science and
Electrical Engineering

Date Approved: ______________________________

ABSTRACT

Title of Thesis: Reclustering for Large Plasticity in Clustered Shape Matching

Michael Falkenstein, Computer Science, 2017

Thesis directed by: Adam W. Bargteil, Assistant Professor
Department of Computer Science and
Electrical Engineering

In this thesis, we present a novel contribution to the clustered shape matching framework. Clustered shape matching describes an algorithm introduced a decade ago by Müller and colleagues (Müller *et al.* 2005a), which was designed to allow for deformable bodies to behave in a physically plausible way when it comes to collisions, deformations, or fractures. This is accomplished by sampling the given deformable object with particles and clustering the particles together in such a way that they can accurately reflect plastic and elastic deformations in the object. These sampled particles determine the degrees of freedom present within the object. At each timestep, a best-fit rigid transformation of the rest of the shape of the object to the current configuration of particles is computed, and Hookean springs are used to pull the particles towards the transformed shape. Clustered shape matching algorithms of this nature have proven to be robust enough to offer realistic physical simulations, while also being efficient enough to run in real-time, offering a level of interactivity for the user. One limitation of clustered shape matching becomes apparent during large plastic deformations. Recently, there was a piece of research work published that extended basic clustered shape matching in an attempt to address this limitation by dynamically adding and removing clusters and particles. In this thesis, we re-visit this limitation and propose a more careful, principle-driven solution to the problem of reclustering. Additionally, we show through experimentation that our proposed solution does not change the behavior of the material of the sampled object. Furthermore, we demonstrate that the

particle reclustering is sufficient in our framework to handle extremely large plastic deformations, allowing us to easily conserve the mass of the object. Lastly, we present a concrete example, highlighting an error in estimating rotations in the original shape matching work of Müller and colleagues (Müller *et al.* 2005a) that has persisted for over a decade through other followup work in shape matching.

Reclustering for Large Plasticity in Clustered Shape Matching

by

Michael Falkenstein

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Computer Science
2017

To The Call, and to Nubby

ACKNOWLEDGMENTS

I would like to thank my thesis adviser, Adam Bargteil, whose door was alwats open whenever I ran into trouble in my research. He allowed my paper to be my own work, but provided guidance whenever necessary. I would also like to thank Caroline, for the constant support through my time at UMBC, and above all, I'd like to thank my parents for giving me this amazing education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Shape matching is a geometrically motivated technique for animating deformable bodies introduced a decade ago by Müller and colleagues (Müller *et al.* 2005b). 1.1 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 (Müller *et al.* 2005b), 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 (Chentanez, Müller, & Macklin 2016a) 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.* 2016a) to fail. In this paper, we revisit

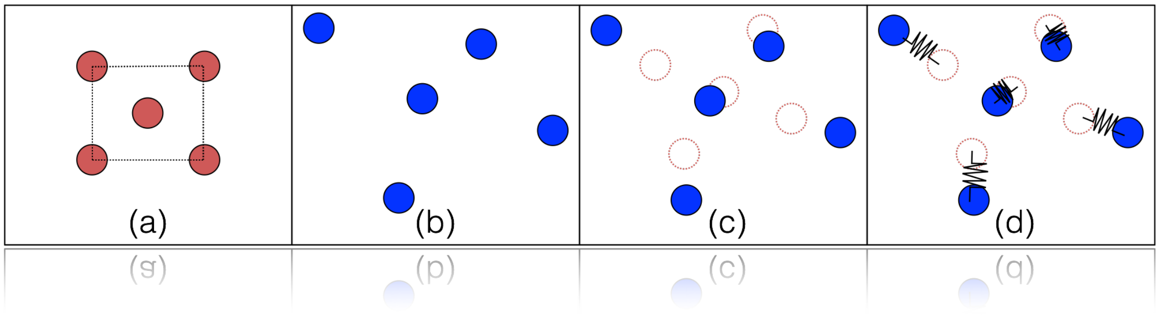

FIG. 1.1. Shape Matching Overview: (a) An object (here, a square) is sampled with particles, p_i, to get rest positions, $\mathbf{r}_i$. (b) As particles are subjected to external forces and constraints, their positions, $\mathbf{x}_i$, are updated in world space. (c) The best-fitting rigid transformation 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.

this problem and propose an alternative approach to reclustering 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.* 2005b) that has been repeated in much of the follow up work.

Chapter 2

BACKGROUND

In this section, we will first review the relevant related work and provide a brief overview of the clustered shape matching technique before detailing my reclustering approach in later sections.

2.1 Related Work

The technique known as shape matching is a geometrically motivated approach to simulating plastic and elastic deformations in objects that was first proposed by (Müller *et al.* 2005a). Through the use of the shape matching approach, Müller and colleagues were able to achieve impressive final results, describing the key advantages of using shape matching: efficiency, stability, and controllability. Their examples demonstrated that the shape matching simulation could be interactively run in real-time, allowing the user to provide input to the simulation while it was running. Even during simulations with many particles, clusters, and forces, the shape matching system was able to remain stable, consistently producing physically viable results. Given these advantages, shape matching was especially appealing to those interested in interactive animation contexts such as video games or other interactive 3D graphics, as it could create a deformable, manipulatable physical body that could be accurately simulated without impacting performance. Müller and colleagues also introduced several useful extensions to the simple shape matching system, including linear and quadratic deformation solvers, rigid deformations, cluster-based deformation, and plastic deformations. These extensions further improved the flexibility and stability of the original shape matching solver.

Two years later, Rivers and James (Rivers & James 2007a) introduced a lattice-based shape matching technique, which used a set of hierarchical 3D lattices to define the shape matching clusters. In their work, they described taking advantage of the regular structure of the lattices in order to achieve extremely high performance, being able to run in real time even on very large, complicated objects. The following year, Steinemann (Steinemann, Otaduy, & Gross 2008a) adopted the lattice-based approach proposed by Rivers and James and altered it slightly such that it could support octrees, enabling spatial adaptivity. This technique relied upon hierarchical sampling and interval-based region definition in order to obtain an accurate, stable definition of the 3D model. Even more recently, Bargteil and Jones (Bargteil & Jones 2014a) further extended the shape matching paradigm by incorporating strain limiting. In their work, they describe how, by limiting the strain of each particle to a pre-determined ceiling, their approach could reduce to explicit integration under small particle strains, while still remaining stable even in the presence of nonlinearities. In a follow-up work published the next year, Jones (Jones *et al.* 2015a) implemented im-

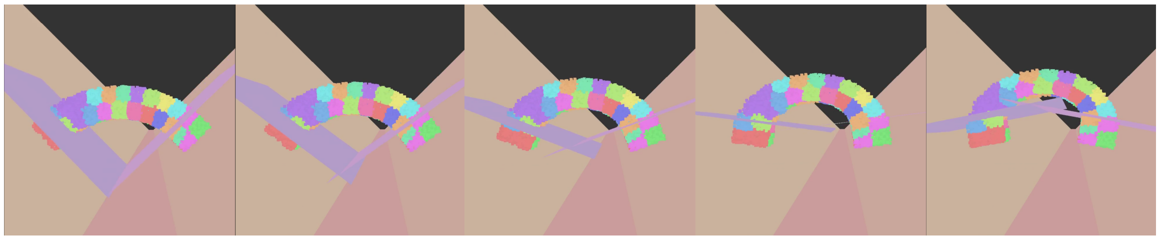

FIG. 2.1. A cube is stretched between two planes. The planes slowly rotate away from each other, but the cube stretches, adding more particles into undersampled areas, maintaining stability without fracturing.

proved particle clustering strategies, including “fuzzy” clustering, in which particles are not strictly limited to belonging to one cluster, but instead could belong to many clusters simultaneously, offering a portion of their physical properties to each. This new technique allowed for greater stability, as well as introduced simple collision proxies for the particle clusters, further increasing efficiency. Jones and colleagues (Jones *et al.* 2016b) again later revisited the clustered shape matching system in order to implement ductile fracturing, such that sampled objects could be deformed significantly before breaking, creating a visually appealing and physically viable stretch, followed by a break.

In order to improve stability for non-volumetric objects, such as seashells or narrow strands of cloth, as well as to simplify sampling, Müller and Chentanez (Müller & Chentanez 2011a) extended the basic shape matching system in order track particle orientation in addition to other physical characteristics. This allowed for particle sampling to remain stable when animating extremely thin objects, such as clothing or hair that is attached to an animated character. The improved particle orientation tracking maintained the efficiency of previous solutions, while extending its applications to these non-volumetric 3D objects. Most recently, Chentanez and colleagues (Chentanez, Müller, & Macklin 2016b) introduced a method of re-sampling objects as they become too extremely deformed. The goal of this work was to address the issue of under-sampling of areas as they become heavily warped. For example, if a an object stretches too far beyond its rest position, the area in which it is stretched could have too few particles to accurately simulate the object’s properties. Chentanez and colleagues offered a technique for re-sampling of the object as well as re-clustering in order to address this issue at runtime. The result was a stable, efficient shape matching extension that could simulate heavily deformed bodies, but made no attempt to conserve mass within these deformed regions of the object.

2.1.1 Clustered Shape Matching

In the interest of keeping this thesis as self-contained as possible, this section will provide a brief overview of the shape matching approach that was originally proposed by Müller and colleagues (Müller *et al.* 2005a), as well as the relevant extensions created by Bargteil, Jones, and colleagues (Bargteil & Jones 2014a; Jones *et al.* 2015a; 2016b) before introducing the new plasticity and fracture models. Finally, we will briefly discuss our approaches to sampling and clustering in the context of the shape matching algorithm.

2.1.2 Shape Matching

In the shape matching framework, continuous, physical objects must first be discretized into a finite set of particles, $p_i \in \mathcal{P}$, with masses, m_i, and rest positions, $\mathbf{r}_i$, that follow a path, $\mathbf{x}_i(t)$. This path is defined in real-world space as a function of time. Shape matching takes its name from the idea that, on each computational step, we match the original rest shape to the current, deformed shape by finding the least-squares best-fit rigid transformation from the rest pose the the deformed pose. This is accomplished by solving for the rotation matrix $\mathbf{R}$ and translation vector $\bar{\mathbf{x}} - \bar{\mathbf{r}}$ that minimizes the equation

$$\sum_i m_i \|\mathbf{R}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) - \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\|^2. \tag{2.1}$$

The best translation is therefore simply given by the difference of the center-of-mass in rest ($\bar{\mathbf{r}}$) and world ($\bar{\mathbf{x}}$ space. This translation is simple, fast, and straightforward to compute. However, computing the rotation matrix $\mathbf{R}$ is considerably more complex and involved. Firstly, the least-squares best-fit linear deformation gradient $\mathbf{F}$ must be computed. More specifically, we seek the $\mathbf{F}$ that minimizes the equation

$$\sum_i m_i \|\mathbf{F}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) - \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\|^2. \tag{2.2}$$

Setting the derivative of this equation with respect to $\mathbf{F}$ to 0 and re-arranging the terms, the equation can be rewritten as

$$\mathbf{F} = \left(\sum_i m_i \mathbf{O}(\mathbf{x}_i, \mathbf{r}_i)\right)\left(\sum_i m_i \mathbf{O}(\mathbf{r}_i, \mathbf{r}_i)\right)^{-1} = \mathbf{A}_{xr}\mathbf{A}_{rr}^{-1}, \tag{2.3}$$

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

$$\mathbf{O}(\mathbf{a}_i, \mathbf{b}_i) = \left(\mathbf{a}_i - \bar{\mathbf{a}}\right)\left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^T. \tag{2.4}$$

For this thesis, we will use $\mathbf{A}_{**}$ is a convenient shorthand. From here, we can compute $\mathbf{R}$ using the polar decomposition

$$\mathbf{F} = \mathbf{RS} = \left(\mathbf{U}\mathbf{V}^T\right)\left(\mathbf{V}\mathbf{\Sigma}\mathbf{V}^T\right) \tag{2.5}$$

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 $\mathbf{F}$, our least-squares best-fit linear deformation gradient. While several other research works (e.g. (Rivers & James 2007a)) have noted that such a polar decomposition can be computed more efficiently than the single value decomposition, especially when warm started, we opt for the SVD for its robustness. It also will be used later in our plasticity model.

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

$$\mathbf{g}_i = \mathbf{R}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) + \bar{\mathbf{x}}. \tag{2.6}$$

Hookean springs are then simulated to create forces that pull the particles towards their goal positions.

2.1.3 Clustered Shape Matching

The motivation behind discretizing a continuous object into particles, and then breaking those particles into clusters is that it allows for rich, physically accurate, localized deformations of the object. Fortunately, working with a large number of clusters of particles is relatively straightforward, despite the complication that we allow a single particle to belong to more than one cluster. When computing a particle's contribution to any given cluster property, the particle's mass is divided amongst the clusters of which it is a member. More specifically, for any given particle p_i in a cluster $c \in \mathcal{C}$, we introduce a weight w_{ic} that describes for much of p_i's mass contributes to the cluster c. From here, m_i is replaced with $w_{ic}m_i$ in equations (2.16) - (2.18) when computing mass and center-of-mass for a cluster. So, if a particle p_i belongs to n_i clusters, then the center-of-mass of a cluster c, $\bar{\mathbf{x}}_c$, is$\mathbf{F}_p = 0$.

$$\bar{\mathbf{x}}_c = \frac{\sum_{p_i \in \mathcal{P}_c} (w_{ic} m_i) \, \mathbf{x_i}}{\sum_{p_i \in \mathcal{P}_c} (w_{ic} m_i)}, \tag{2.7}$$

where $\mathcal{P}_c$ is the set of all particles that belong to cluster c. Furthermore, when computing the goal position of particle p_i, denoted as $\mathbf{g}_i$, we perform a weighted average of the goal position given by each cluster to which is belongs. This weighted average is computer as

$$\mathbf{g}_i = \sum_c w_{ic} \, \mathbf{g}_{ic}, \tag{2.8}$$

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

2.1.4 Clustering

For this thesis work, we use the clustering method originally proposed by Jones and colleagues (Jones *et al.* 2015a). The proposed method is a slight variation of the fuzzy c-means clustering algorithm, whose goal is to produce overlapping clusters where each member particle may belong to several clusters simultaneously, to varying degrees. Similar to the well-known k-means clustering technique, fuzzy c-means alternates between updating the particle-cluster membership relationship and updating each cluster's center-of-mass. In this scenario, updating cluster membership involves also updating the weights, w_{ic}, as well as the cluster centers-of-mass, which are computed using the weighted center-of-mass of each member particle. In their work, Jones and colleagues discuss several different weighting functions, the affect of cluster size, the degree of overlap, as well as other factors to consider. In their work, the authors settle on the fuzzy c-means weighting function defined by (Dunn 1973)

$$\mathrm{fcm}(\mathbf{x}_i, \mathbf{x}_c, h) = \frac{1}{\sum_{d \in \mathcal{C}} \left(\frac{\|\mathbf{x}_i - \mathbf{x}_c\|}{\|\mathbf{x}_i - \mathbf{x}_d\|} \right)^{\frac{2}{m-1}}} \tag{2.9}$$

Additionally, the authors experimented on a range of cluster sizes, ranging from 10 to 50 particles. Lastly, the degree of overlap was also explored. The conclusion of this work was that all three have significant impact on the end behavior of the material, and each must be chosen manually in order to find the optimal fit for each simulation.

Strain Limiting To maintain stability during the simulation's runtime, we include the strain limiting approach proposed and advocated by Bargteil and Jones (Bargteil & Jones 2014a). However, in this implementation we also include large plastic deformations. In the presence of extreme plastic warping, as proposed in their work, typically the maximum allowed stretch (γ in their work) is increased. This was found experimentally to improve stability of the simulation when two adjacent clusters create opposing forces as they disagree about the current rest shape.

Collision Handling Again, we leverage the collision handling approach proposed by Jones and colleagues (Jones *et al.* 2015a) for fast, accurate collision solving. As discussed in their work, collision proxies are created using spheres intersected with half-spaces for each cluster. This approach is very appropriate to this application, as the fracture approach we use functions by dividing the clusters in an object along 3D planes. This approach works because we include the distance constraint during clustering. That is, a cluster is defined by user input as particles within a certain distance of the cluster's center. This means a cluster's shape can be generally well approximated by spheres. This approach works well in volumetric objects, but can produce poor results in the case of thin sheets or narrow strands of clothing. The solution proposed by Jones and colleagues was to supplement the sphere approximation with plane approximation for sufficiently thin objects. We also use this technique to create a robust but easily simulated collision geometry.

Sampling Geometry Since the shape matching technique begins by discretizing a single object into many particles, the approach to the particle sampling has a considerable impact on the the resulting simulation. Typically, geometry sampling can either be done on a regular grid, or randomly, usually following a noise scheme. The regular grid heuristic has the advantage of being very fast and stable for geometric objects, with few faces and sharp angles, such as cubes, pyramids, and other euclidean solids. Using blue noise to generate particle locations has the advantage of being much more flexible, as it can more easily capture curves or small angles of an object. Through experimentation, we found a blue noise sampling technique produced the best results. The blue noise sampler that we implemented is based on Bridson's fast Poisson disk sampling (Bridson 2007a).

For objects whose boundaries are defined by an arbitrary manifold, we simply sample particles withing the bounding box of the object, and discard particles that are found to be outside the object's surface.

Plasticity The approach to plastic deformation used in this thesis adapts the model proposed by Bargteil and colleagues (Bargteil *et al.* 2007) to fit into the clustered shape matching framework. To accomodate plastic deformation, we store and update an addition matrix for each cluster c, denoted as $\mathbf{F}^p_c$. In the interest of readability, we drop the subscript, but the following equation is computer for each cluster in the system

$$\mathbf{F}^e = \mathbf{F}\left(\mathbf{F}^p\right)^{-1}, \tag{2.10}$$

where $\mathbf{F}$ is given by(2.18). we then decompose $\mathbf{F}^e$ in (2.20). $\mathbf{F}^p$ is initialized as the identity matrix $\mathbf{I}$, then each computational timestep we compute the volume preserving part of the diagonalized $\mathbf{F}^e$ as

$$\mathbf{F}^* = \det(\mathbf{\Sigma}^e)^{-1/3}\mathbf{\Sigma}^e. \tag{2.11}$$

From here, we compare

$$||\mathbf{F}^* - \mathbf{I}||_F \tag{2.12}$$

to a pre-determined plastic yield threshold, denoted λ, where $||\cdot||_F$ is the Frobenius norm. If this threshold is not exceeded, then $\mathbf{F}^p$ remains unchanged on this step. However, if this threshold is exceeded, then $\mathbf{F}^p$ is updated using the equation

$$\mathbf{F}^p_{new} = \left(\mathbf{F}^*\right)^{\gamma}\mathbf{V}\mathbf{F}^p_{old}, \tag{2.13}$$

where $\mathbf{V}$ is the matrix of right singular vectors in (2.20) and γ is computer as

$$\gamma = \min\left(\frac{\nu * \|\mathbf{F}^* - \mathbf{I}\|_F - \lambda - K\alpha}{\|\mathbf{F}^* - \mathbf{I}\|_F}, 1\right), \tag{2.14}$$

where ν and K are user-determined flow rate and work hardening/softening constants, respectively. Also here, α is a measure of cumulative stress that is initialized to zero, and then updated using the equation

$$\dot{\alpha} = \|\mathbf{F}^e - \mathbf{I}\|_F. \tag{2.15}$$

Typically, additional left-hand rotations would be applied during these calculations. However, since we perform a polar decomposition in (2.20), these rotations would be discarded, and so there is no need to compute them here.

2.1.5 Clustered Shape Matching

For completeness and readability, we first briefly review the shape matching approach of Müller and colleagues (Müller *et al.* 2005b) and the extensions of Bargteil, Jones, and colleagues (Bargteil & Jones 2014b; Jones *et al.* 2015b; 2016a) before introducing our plasticity and fracture models. Finally we briefly discuss our approaches to sampling and clustering.

2.1.6 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

$$\sum_i m_i \|\mathbf{R}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) - \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\|^2. \tag{2.16}$$

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

$$\sum_i m_i \|\mathbf{F}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) - \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\|^2. \tag{2.17}$$

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

$$\mathbf{F} = \left(\sum_i m_i \mathbf{O}(\mathbf{x}_i, \mathbf{r}_i)\right)\left(\sum_i m_i \mathbf{O}(\mathbf{r}_i, \mathbf{r}_i)\right)^{-1} = \mathbf{A}_{xr}\mathbf{A}_{rr}^{-1}, \tag{2.18}$$

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

$$\mathbf{O}(\mathbf{a}_i, \mathbf{b}_i) = \left(\mathbf{a}_i - \bar{\mathbf{a}}\right)\left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^T, \tag{2.19}$$

and $\mathbf{A}_{**}$ is a convenient shorthand.

We then compute $\mathbf{R}$ using the polar decomposition,

$$\mathbf{F} = \mathbf{R}\mathbf{S} = \left(\mathbf{U}\mathbf{V}^T\right)\left(\mathbf{V}\mathbf{\Sigma}\mathbf{V}^T\right) \tag{2.20}$$

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 & James 2007b)) 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 2.1.6).

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

$$\mathbf{g}_i = \mathbf{R}\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) + \bar{\mathbf{x}}. \tag{2.21}$$

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 (2.16)-(2.18) 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

$$\bar{\mathbf{x}}_c = \frac{\sum_{p_i \in \mathcal{P}_c} (w_{ic} m_i)\, \mathbf{x_i}}{\sum_{p_i \in \mathcal{P}_c} (w_{ic} m_i)}, \tag{2.22}$$

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,

$$\mathbf{g}_i = \sum_c w_{ic}\, \mathbf{g}_{ic}, \tag{2.23}$$

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 (Jones *et al.* 2015b). 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 (Jones *et al.* 2015b) 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 (Bargteil & Jones 2014b). However, in the presence of plastic deformation (see 2.1.6) 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 (Jones *et al.* 2015b) 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 2007b). 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 (Bargteil *et al.* 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

$$\mathbf{F}^e = \mathbf{F}\left(\mathbf{F}^p\right)^{-1}, \tag{2.24}$$

where $\mathbf{F}$ is given by 2.18. We then decompose $\mathbf{F}^e$ as in 2.20.

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

$$\mathbf{F}^* = \det(\mathbf{\Sigma}^e)^{-1/3}\mathbf{\Sigma}^e. \tag{2.25}$$

We then compare

$$||\mathbf{F}^* - \mathbf{I}||_F \tag{2.26}$$

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

$$\mathbf{F}^p_{new} = \left(\mathbf{F}^*\right)^{\gamma}\mathbf{V}\mathbf{F}^p_{old}, \tag{2.27}$$

where $\mathbf{V}$ is the matrix of right singular vectors in 2.20 and γ is given by

$$\gamma = \min\left(\frac{\nu * ||\mathbf{F}^* - \mathbf{I}||_F - \lambda - K\alpha}{||\mathbf{F}^* - \mathbf{I}||_F}, 1\right), \tag{2.28}$$

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

$$\dot{\alpha} = ||\mathbf{F}^e - \mathbf{I}||_F. \tag{2.29}$$

We do not apply additional left-hand rotations when computing $\mathbf{F}^p_{new}$ as these would be discarded during the decomposition in 2.20.

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,

$$\mathbf{F}^e = \mathbf{A}_{xr}\mathbf{A}_{rr}^{-1}\left(\mathbf{F}^p\right)^{-1} \tag{2.30}$$

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

$$\mathbf{g}_i = \mathbf{R}\mathbf{F}^p\left(\mathbf{r}_i - \bar{\mathbf{r}}\right) + \bar{\mathbf{x}}. \tag{2.31}$$

Regrettably, these details were omitted by Jones and colleagues (Jones *et al.* 2016a).

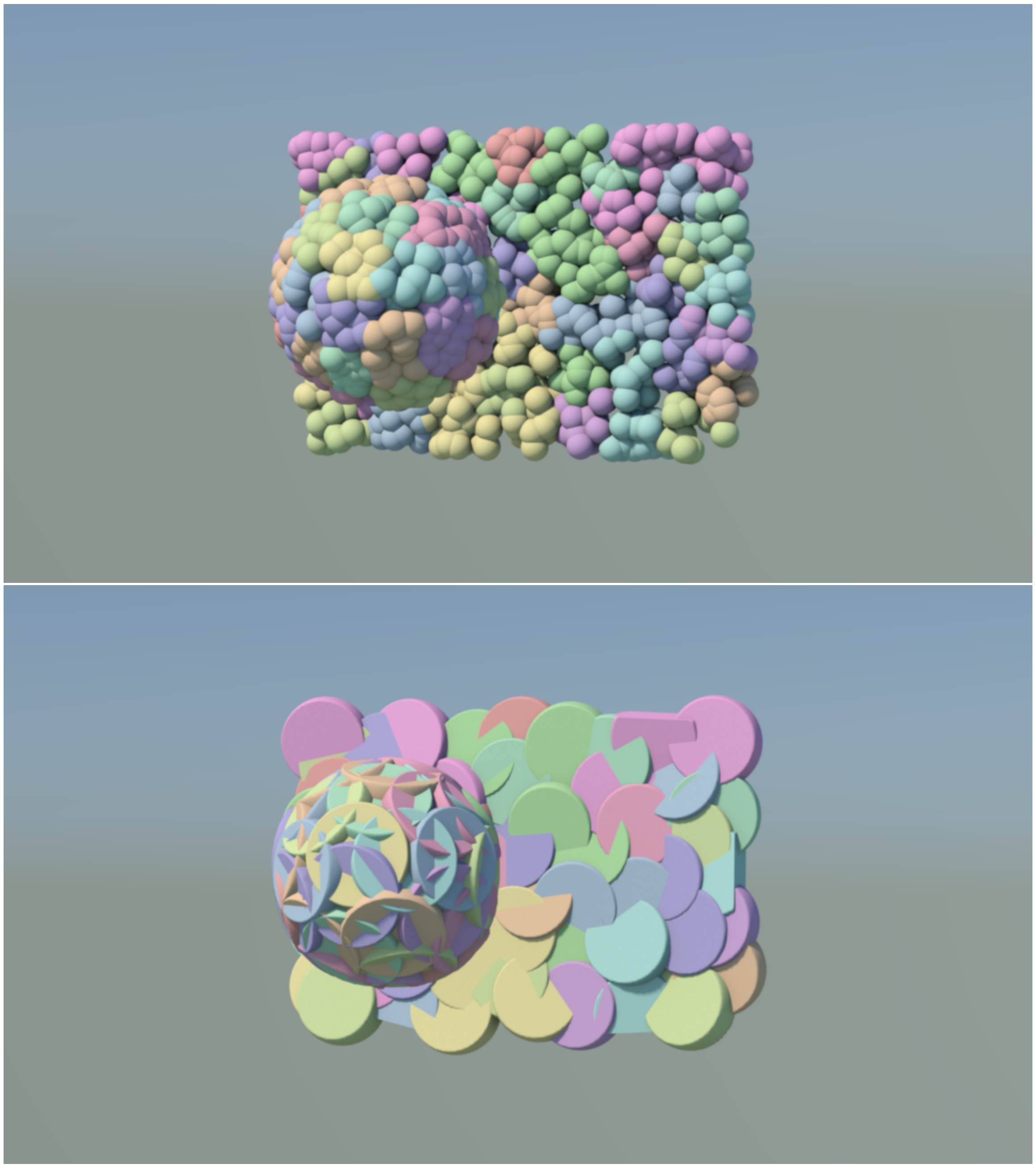

FIG. 2.2. Top: The particles in one of our scenes color-coded by the closest cluster. Bottom: The collision geometry for the same scene.

Chapter 3

METHODS

3.0.1 The Best Rotation

We begin by explicitly highlighting an error in early shape matching work. Mueller and colleagues (Müller *et al.* 2005b) 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 **F** if $\mathbf{A}_{rr}$ is diagonal or **F** has a condition number of 1. However, if $\mathbf{A}_{rr}$ is not diagonal *and* **F** includes a non-uniform scale, the polar decompositions of $\mathbf{A}_{xr}$ and **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

$$\mathbf{A}_{rr} = \begin{pmatrix} 20 & 12 \\ 12 & 20 \end{pmatrix}^{-1} = \frac{1}{64}\begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix}. \tag{3.1}$$

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

$$\mathbf{A}_{xr} = \begin{pmatrix} 40 & 24 \\ 12 & 20 \end{pmatrix}, \tag{3.2}$$

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

$$\mathbf{R} = \begin{pmatrix} .9806 & -.19611 \\ .19611 & .9806 \end{pmatrix}^{-1}, \tag{3.3}$$

when in fact the transformation contained no rotation.

$$\mathbf{F} = \mathbf{A}_{xr}\mathbf{A}_{rr} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \tag{3.4}$$

recovers the transformation. This error has persisted in the shape matching literature (Rivers & James 2007b; Steinemann, Otaduy, & Gross 2008b; Müller & Chentanez 2011b; Choi 2014). 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 diagonal and 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 (Choi 2014) and adopted by Chentanez and colleagues (Chentanez, Müller, & Macklin 2016a). Consequently we adopt the simpler plasticity model of Jones and colleagues (Jones *et al.* 2016a), 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 $\mathbf{F}^e$.

3.0.2 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. 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 (Chentanez, Müller, & Macklin 2016a) adopted the same storage.

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 (Bargteil *et al.* 2007) who globally remeshed whenever an individual tetrahedron's condition number exceeded a threshold, but diverges from the approach of Chentanez and colleagues (Chentanez, Müller, & Macklin 2016a) 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 2.17 gradually decreases. Changing weights shifts cluster centers, which requires updating relative positions of cluster members to keep the center of mass at the origin.

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 (Chentanez, Müller, & Macklin 2016a) 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 (Bargteil *et al.* 2007). As pointed out by Wicke and colleagues (Wicke *et al.* 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 (Bargteil *et al.* 2007). More recently, Jones and colleagues (Jones *et al.* 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,

$$\arg\min_{\mathbf{e}_i,\mathbf{e}_c} \sum_{i,c} w_{ic} \|\mathbf{R}\mathbf{F}_c^p \mathbf{p}_{ic} - (\mathbf{e}_i - \mathbf{e}_c)\|^2. \tag{3.5}$$

To optimize 3.5 we use the strain limiting approach of Bargteil and colleagues (Bargteil & Jones 2014b) 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 3.5 as Jones and colleagues (Jones *et al.* 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 3.5 was roughly two orders of magnitude larger than our nonlinear optimization.

Local Embedding Jones and colleagues (Jones *et al.* 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

$$\mathcal{C}_e = \mathcal{C}_p \cup \{c \in \mathcal{C} \mid (\exists d \in \mathcal{C}_p)\,(\exists q \in c)\,[q \in d]\} \tag{3.6}$$

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

$$\mathcal{P}_e = \{q \in \mathcal{P} \mid (\exists c \in \mathcal{C}_e)\,[q \in c]\}. \tag{3.7}$$

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 2.1.6) 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 (Chentanez, Müller, & Macklin 2016a) 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.

Chapter 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, but not identical to Chentanez and colleagues (Chentanez, Müller, & Macklin 2016a), because we do not compute an estimate for $\mathbf{F}^e$ from neighboring clusters. The final frame from these examples using our approach are shown in figure 4.1. Without optimization the embedding space or the cluster location the simulation again goes unstable. Quantitative results using our unoptimized research code are given in 4.1. 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

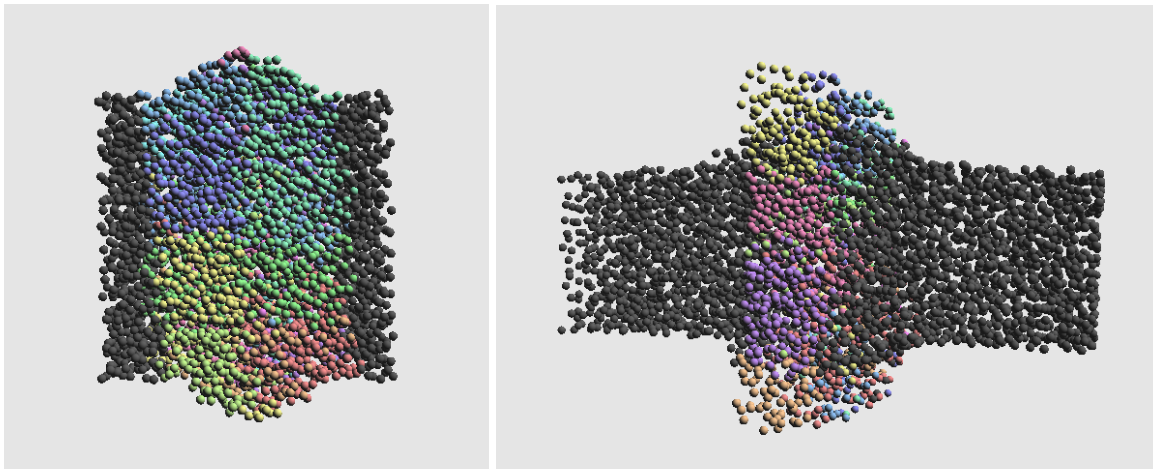

FIG. 4.1. A beam is compressed. Left: A highly plastic beam. Right: More elastic deformation. Colors indicate the closest cluster center. Grey particles are in their original clusters.

Table 4.1. Timing results in ms per frame taken on a Macbook Pro with a 2.5 Ghz Intel i7 processor.

Example	# Particles	# Clusters	Dynamics	Plasticity	Reclustering	Total
Beam (no reclustering)	5317	200	2.07	0.07	0	16.17
Beam (our algorithm)	5317	200	2.13	0.07	19.12	38.21
Beam (no optimization)	5317	200	0.65	0.09	19.64	37.61
Beam (our algorithm)	5317	200	0.49	0.04	3.65	14.17

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 5.

In 4.3 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.

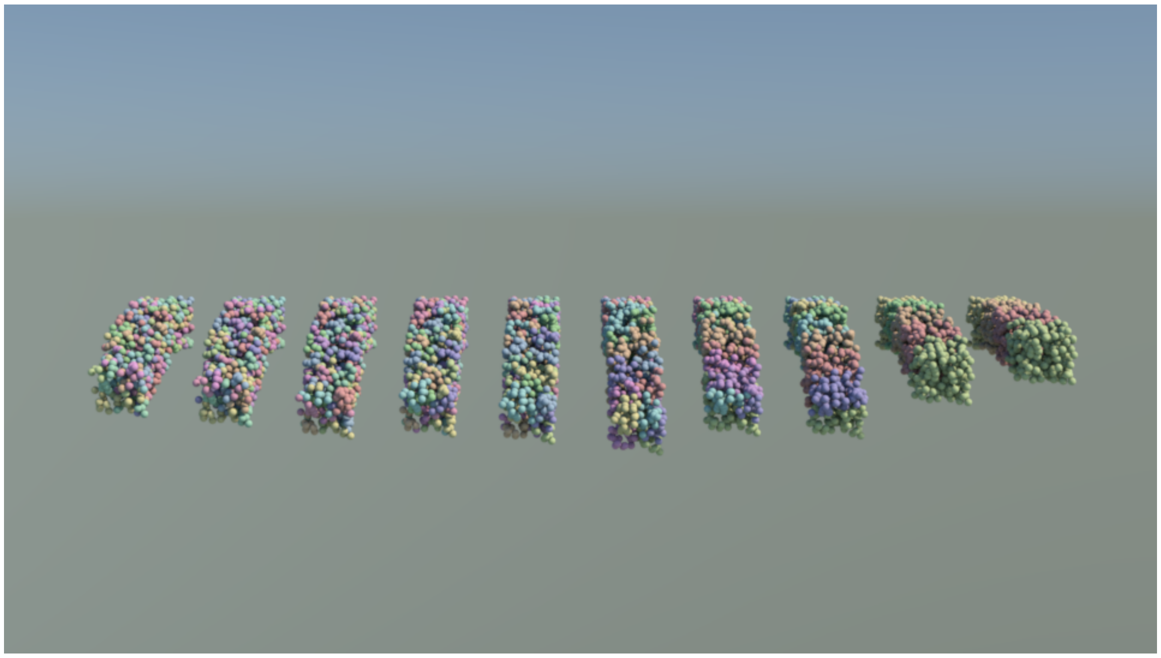

FIG. 4.2. Decreasing number of clusters from left to right. As the number of clusters decreases the apparent stiffness increases. The leftmost bars are visually almost indistinguishable.

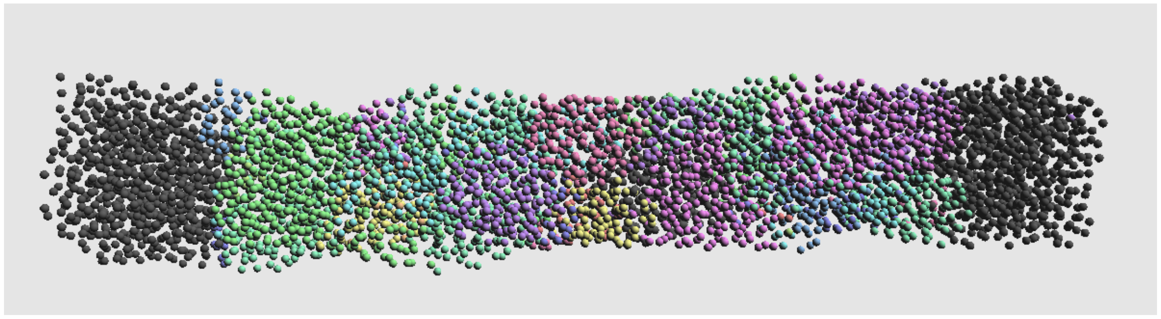

FIG. 4.3. A twisted plastic beam. Colors indicate the closest cluster center. Grey particles are in their original clusters.

Limitations and Future Work A limitation of our approach when compared to Chentanez and colleagues (Chentanez, Müller, & Macklin 2016a) 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 (Bhattacharya, Gao, & Bargteil 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.

Chapter 5

CONCLUSION

One of the primary advantages of using the clustered shape matching framework is that the number of degrees of freedom far exceeds the number of "integration units" – clusters in this case. The opposite is true of the finite element methods with unstructured meshes where the number of tetrahedra is often considerably larger than the number of vertices in the simulation. For graphical applications, visual detail, which correlates with the number of degrees of freedom, is one of principal importance, and computation, which correlates with the number of clusters, is often times limited. For these reasons, the clustered shape matching framework is extremely appealing in the field of computer animation and interactive 3D graphics. The utility and versatility of this framework is greatly improved upon by our extensions to clustering, reclustering, and object resampling.

In this thesis, we have extended the clustered shape matching framework for animating deformable boding by introducing a novel method of reclustering and resampling of the original object geometry. This extension has been shown to be highly flexible and versatile, producing results that accurately simulate the physical properties of the object, even under extreme deformations, where an object might be nearly completely flattened, or stretched beyond all recognition. This implementation remains stable and efficient even in these more extreme tests. This new implementation improves both the power and versatility of the existing clustered shape matching framework. We have also highlighted a mathematical error that has persisted for over a decade through many follow-up works on the original clustered shape matching publication.

REFERENCES

[1] Bargteil, A. W., and Jones, B. 2014a. Strain limiting for clustered shape matching. In *Proceedings of the Seventh International Conference on Motion in Games*, 177–179. ACM.

[2] Bargteil, A. W., and Jones, B. 2014b. Strain limiting for clustered shape matching. In *Proceedings of the Seventh International Conference on Motion in Games*, 177–179.

[3] 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.

[4] Bhattacharya, H.; Gao, Y.; and Bargteil, A. W. 2011. A level-set method for skinning animated particle data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

[5] Bhattacharya, H.; Gao, Y.; and Bargteil, A. W. 2015. A level-set method for skinning animated particle data. *IEEE Trans. Vis. Comput. Graph.* 21:315–327.

[6] Bridson, R. 2007a. Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH sketches*, 22.

[7] Bridson, R. 2007b. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*.

[8] Chentanez, N.; Müller, M.; and Macklin, M. 2016a. Real-time simulation of large elasto-plastic deformation with shape matching. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 159–167.

[9] Chentanez, N.; Müller, M.; and Macklin, M. 2016b. Real-time simulation of large elasto-plastic deformation with shape matching.

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

[11] Dunn, J. C. 1973. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics* 3(3):32–57.

[12] 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):21:1–21:9.

[13] Jones, B.; Martin, A.; Levine, J. A.; Shinar, T.; and Bargteil, A. W. 2015a. Clustering and collision detection for clustered shape matching. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, 199–204. ACM.

[14] Jones, B.; Martin, A.; Levine, J. A.; Shinar, T.; and Bargteil, A. W. 2015b. Clustering and collision detection for clustered shape matching. In *Proceedings of ACM Motion in Games*.

[15] Jones, B.; Martin, A.; Levine, J. A.; Shinar, T.; and Bargteil, A. W. 2016a. Ductile fracture for clustered shape matching. In *Proceedings of the ACM SIGGRAPH symposium on Interactive 3D graphics and games*.

[16] Jones, B.; Martin, A.; Levine, J. A.; Shinar, T.; and Bargteil, A. W. 2016b. Ductile fracture for clustered shape matching. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 65–70. ACM.

[17] Müller, M., and Chentanez, N. 2011a. Solid simulation with oriented particles. In *ACM transactions on graphics (TOG)*, volume 30, 92. ACM.

[18] Müller, M., and Chentanez, N. 2011b. Solid simulation with oriented particles. *ACM Trans. Graph.* 30(4):92:1–92:10.

[19] Müller, M.; Heidelberger, B.; Teschner, M.; and Gross, M. 2005a. Meshless deformations based on shape matching. *ACM transactions on graphics (TOG)* 24(3):471–478.

[20] Müller, M.; Heidelberger, B.; Teschner, M.; and Gross, M. 2005b. Meshless deformations based on shape matching. *ACM Trans. Graph.* 24(3):471–478.

[21] Müller, M.; Heidelberger, B.; Hennix, M.; and Ratcliff, J. 2007. Position based dynamics. *J. Vis. Comun. Image Represent.* 18(2):109–118.

[22] Rivers, A. R., and James, D. L. 2007a. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics (TOG)* 26(3):82.

[23] Rivers, A. R., and James, D. L. 2007b. Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.* 26(3).

[24] Steinemann, D.; Otaduy, M. A.; and Gross, M. 2008a. Fast adaptive shape matching deformations. In *Proceedings of the 2008 ACM SIGGRAPH/eurographics symposium on computer animation*, 87–94. Eurographics Association.

[25] Steinemann, D.; Otaduy, M. A.; and Gross, M. 2008b. Fast adaptive shape matching deformations. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 87–94.

[26] Wicke, M.; Ritchie, D.; Klingner, B. M.; Burke, S.; Shewchuk, J. R.; and O'Brien, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on Graphics* 29(4):49:1–11.