Multi-resolution Clustering for Enhanced Elastic Behavior in Clustered Shape Matching

Aditya Viswanathan Kaliappan
adityak1@umbc.edu
University of Maryland, Baltimore County (UMBC)
Baltimore, Maryland, USA

Adam W. Bargteil
adamb@umbc.edu
University of Maryland, Baltimore County (UMBC)
Baltimore, Maryland, USA

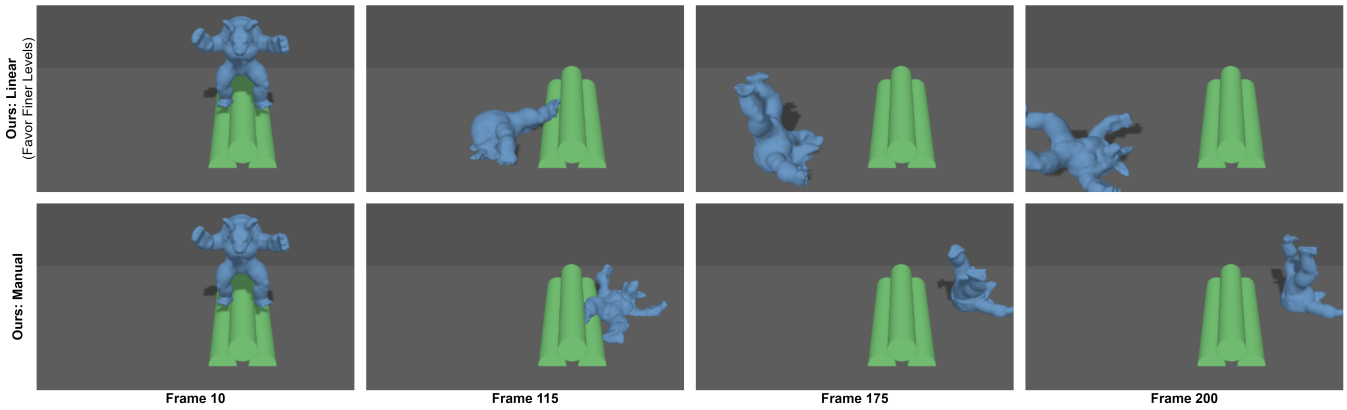

Figure 1: Frames from Armadillo Animation.

ABSTRACT

Clustered shape matching is an approach for physics-based animation of deformable objects, which breaks an object into overlapping clusters of particles. At each timestep, it computes a best-fit rigid transformation between a cluster's rest state and current particle configuration and Hookean springs are used to pull particles toward desired goal positions. In this paper, we present multi-resolution clustering as an extension to clustered shape matching. We iteratively construct fine-to-coarse sets of clusters and weights over the set of particles and compute dynamics in a single coarse-to-fine pass. We demonstrate that our approach enhances the possible elastic behavior available to artists and provides an intuitive parameterization to blend between stiffness and deformation richness, which are in contention in the traditional clustered shape matching approach that operates at a single spatial scale. We can specify a different stiffness value for each resolution level, where a greater weight at coarser levels result in a stiffer object while a greater weight at finer levels yield richer deformation; we evaluate a number of approaches for choosing these stiffness values and demonstrate the differences in the accompanying video.

MIG '20, October 16–18, 2020, Virtual Event, SC, USA

ACM ISBN 978-1-4503-8171-0/20/10…$15.00
https://doi.org/10.1145/3424636.3426902

CCS CONCEPTS

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

KEYWORDS

shape matching, multi-resolution clustering

ACM Reference Format:
Aditya Viswanathan Kaliappan and Adam W. Bargteil. 2020. Multi-resolution Clustering for Enhanced Elastic Behavior in Clustered Shape Matching. In *Motion, Interaction and Games (MIG '20), October 16–18, 2020, Virtual Event, SC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3424636.3426902

1 INTRODUCTION

A classical problem in physics-based animation is developing techniques and models to simulate the deformation of soft bodies in a scene. These approaches should obey the laws of Newtonian physics, while ensuring stability of objects under deformation.

Shape matching is a geometrically-motivated approach for simulating deformable objects that was introduced by Müller and colleagues [2005]. In this approach, each deformable object in a scene is sampled with particles. At each timestep, a best-fit rigid transformation between each object's rest state and the current configuration of particles is computed. Finally, Hookean springs are used to pull particles towards goal positions, resulting in a transformed shape based on external forces (e.g. gravity) and the object's response to collisions in the environment.

A natural extension, also introduced by Müller and colleagues [2005], involves breaking each object into overlapping clusters of particles; this was coined "clustered shape matching" in later

research. Despite lacking a well-defined mathematical foundation, clustered shape matching provides advantages during simulation, including the ability to use cluster overlap to keep objects from falling apart. Furthermore, as the results by Jones and colleagues [2015] suggest, by increasing the number of overlapping clusters that an object is broken into (and therefore reducing the number of particles contained in any one cluster), a richer object deformation space is produced. However, Falkenstein and colleagues [2017] show that increasing the number of clusters also reduces the apparent stiffness of objects in simulation. Therefore, a reasonable problem to investigate is whether a combination of both coarse and fine sets of clusters in elasticity computations will provide a good blend of the desired stiffness and deformation properties.

The application of multi-resolution clusters in elasticity computations has not been considered previously for clustered shape matching. Thus, we present this as a novel extension to the traditional framework. In this paper, we demonstrate that, while being a straightforward extension of clustered shape matching, our multi-resolution approach also displays the following behaviors:

- By considering coarser resolution levels (small set of large clusters), objects demonstrate increased stiffness not visible in the traditional framework with a large set of small clusters.
- By considering finer resolution levels (large set of small clusters), objects demonstrate richer deformations not apparent in the traditional framework with a small set of large clusters.
- Our approach incurs minimal additional computational cost.
- Our approach provides valuable artistic control of behavior not available with single resolution clusters; it allows artists to fine-tune the balance between stiffness and richness of deformation.
- After experimenting with several automatic approaches for setting the stiffness at different clustering sizes, we found that the framework is not particularly sensitive to the choice of function, but only to the relative weighting between coarse, medium, and fine.

2 RELATED WORK

Müller and colleagues [2005] introduced a geometrically-motivated shape matching approach having several advantages such as efficiency, stability, and controllability. They also introduced several extensions to their approach, including clustered shape matching. More recently, Bargteil and Jones [2014] applied strain limiting to clustered shape matching. Jones and colleagues [2015] advocated a fuzzy c-means clustering approach, which produces a set of overlapping clusters such that each particle may belong to several clusters to varying degrees. They also introduced an approach for collision detection and handling by using the intersection of spheres with half-spaces of planes as collision proxy geometry. Jones and colleagues [2016] introduced an approach for enabling ductile fracture in clustered shape matching. Falkenstein and colleagues [2017] presented a method for reclustering particles to handle very large plastic deformations.

Clustered shape matching is just one of many physics-based animation approaches. For instance, lattice-based shape matching was introduced by Rivers and James [2007], in which they defined a deformable surface mesh as a lattice structure, and constructed overlapping lattice clusters to allow for object deformation. Steinemann and colleagues [2008] adopted a similar approach, but instead used a hierarchical representation based on an octree to achieve improved performance. An increasingly popular approach for real-time animation of deformable objects is position-based dynamics (PBD) [Bender et al. 2014; Müller et al. 2007]. In this approach, "forces" are applied directly on particles by using constraints. However, Bargteil and Jones [2014] note that while constraints can be applied to make PBD stable and useful for real-time animation, it nevertheless violates Newton's first and second laws of motion due to direct positional updates of particles. A variant of PBD is projective dynamics [Bouaziz et al. 2014; Liu et al. 2013], which involves defining a set of constraints (e.g. collision) and projecting a set of points onto the constraint set [Müller et al. 2007]. Other approaches include using frame-based models [Faure et al. 2011; Gilles et al. 2011], where a sparse set of coordinate frames are defined to allow for greater degrees of freedom and physically realistic deformation [Gilles et al. 2011].

A common approach for encoding spatial information is multi-scale clustering. For instance, He and colleagues [2015] used a bottom-up approach to construct a bounding volume hierarchy (BVH) by iteratively clustering sets of triangles in a triangle mesh, such that the number of elementary collision tests performed in animation of topology changing models is reduced, thus improving simulation runtime. Müller [2008] also used a bottom-up approach to construct a hierarchy of particles and constraints for position-based dynamics (PBD), demonstrating faster convergence in solving for PBD constraints while maintaining the ability to handle non-linear constraints as with traditional PBD. Another popular data structure to encode 3D spatial information is an octree; for example, Srihari [1984] presented a method of applying octrees to represent three-dimensional geometry at varying levels of detail.

These prior works all have the common intuition that considering multi-resolution spatial features allows for improved/unique behavior in their corresponding tasks. Furthermore, these works [He et al. 2015; Müller 2008] tend to use a bottom-up approach (i.e. fine-to-coarse pass) to construct each resolution level. While we did consider similar methods for our approach, these methods do not lend themselves well to the use of fuzzy c-means clustering, which Jones and colleagues [2015] demonstrate works well to generate both coarse and fine sets of overlapping clusters for reasonable animation behavior in the clustered shape matching framework. Furthermore, since fuzzy c-means generates particle membership weights based on the distance of particles from cluster centers, there is no obvious way to propagate particle membership weights between resolution levels if, for instance, only a subset of particles are considered when constructing the next coarser resolution level. However, we do not discount the merits of prior multi-resolution clustering approaches in related areas of research [He et al. 2015; Müller 2008]; in fact, our approach also constructs sets of clusters at different resolution levels using a bottom-up approach.

The intuition for the expected behavior of our approach is most similar to some research in computer vision and cloth simulation. For instance, in computer vision, Ma and colleagues [2015] constructed a set of coarse-to-fine convolution features in order to improve visual object tracking. More specifically, their approach enabled them to balance the properties of coarser and finer

features, such that the coarser features capture semantics for high-level visual recognition and the finer features capture details such as object positions. Similarly, in cloth simulation, Kavan and colleagues [2011] presented a multi-resolution approach for upsampling, which attempts to construct an upsampling operator that enhances a coarse cloth mesh with details from more finely wrinkled cloth meshes.

In both works, a multi-resolution approach is used to combine the properties of coarse and fine features. In the context of clustered shape matching, results from prior research [Falkenstein et al. 2017; Jones et al. 2015] demonstrate that an object's apparent stiffness increases with fewer, larger clusters (i.e. "coarse features"), while its deformation space becomes richer with more, smaller clusters (i.e. "fine features"). Given that a multi-resolution approach considering both coarse and fine features works well for computer vision [Ma et al. 2015] and cloth simulation [Kavan et al. 2011], it is reasonable to extend this notion to clustered shape matching to allow for a good blend of desired stiffness and deformation properties.

3 CLUSTERED SHAPE MATCHING

For completeness, we review the formulations for the shape matching and clustered shape matching approaches. Readers familiar with clustered shape matching may safely skip this section.

3.1 Shape Matching

Shape matching is an approach suggested by Müller and colleagues [2005] for physics-based computer animation of soft bodies. A graphical overview of this approach is presented in Figure 2. More formally, an object is broken into a set of particles $\mathcal{P}$. Each particle $i \in \mathcal{P}$ has mass m_i and initial rest position r_i. At each timestep, the approach solves for the rotation matrix, $\mathbf{R}$, and translation vector, $\bar{\mathbf{x}} - \bar{\mathbf{r}}$, that minimizes

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

The name "shape matching" is fitting, since for each animation frame, we try to *match* the *rest shape* to the *deformed shape* by identifying the least-squares best-fit rigid transformation from the rest shape to the deformed shape. This rigid transformation is given by $\mathbf{R}$ and $\bar{\mathbf{x}} - \bar{\mathbf{r}}$, both of which are found by minimizing Equation 1.

Müller and colleagues [2005] identify that the optimal translation vector is given by the shape's center-of-mass in the rest ($\bar{\mathbf{r}}$) and world ($\bar{\mathbf{x}}$) space positions; that is, the optimal translation vector can be expressed as

$$\bar{\mathbf{x}} - \bar{\mathbf{r}} = \mathbf{x}_{cm} - \mathbf{r}_{cm} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i} - \frac{\sum_i m_i \mathbf{r}_i}{\sum_i m_i}. \tag{2}$$

Since computing the optimal rotation $\mathbf{R}$ is more involved, we first relax this problem by finding an optimal linear transformation $\mathbf{A}$. By denoting the relative location vectors $\mathbf{q}_i = \mathbf{r}_i - \bar{\mathbf{r}}$ and $\mathbf{p}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ for each particle with respect to the shape's rest and world center-of-mass, respectively, Equation 1 can be rewritten as

$$\sum_i m_i \|\mathbf{A}\mathbf{q}_i - \mathbf{p}_i\|^2. \tag{3}$$

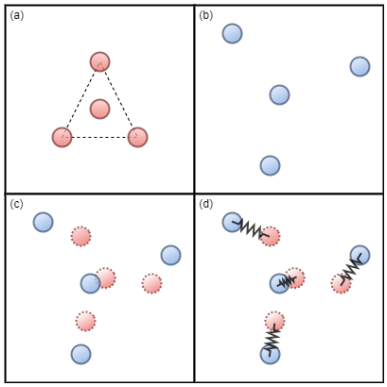

Figure 2: Shape Matching Overview: (a) An object is sampled with a set of particles (in red) to compute rest positions. (b) Particle positions (in blue) are updated in world space as the particles are subjected to external forces and constraints. (c) The particles' goal positions (as dotted red circles) are computed based on the best-fitting rigid transformation of particles from rest to world positions. (d) Hookean springs pull the particles from world positions toward the goal positions.

By minimizing Equation 3, the optimal $\mathbf{A}$ can be computed as

$$\mathbf{A} = \left(\sum_i m_i \mathbf{p}_i \mathbf{q}_i^T\right)\left(\sum_i m_i \mathbf{q}_i \mathbf{q}_i^T\right)^{-1}. \tag{4}$$

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

$$\mathbf{A} = \mathbf{R}\mathbf{S} = \left(\mathbf{U}\mathbf{V}^T\right)\left(\mathbf{V}\Sigma\mathbf{V}^T\right), \tag{5}$$

where $\mathbf{S} = \mathbf{V}\Sigma\mathbf{V}^T$ is a symmetric matrix and $\mathbf{U}\Sigma\mathbf{V}^T$ is the singular value decomposition (SVD) of $\mathbf{A}$.

Given $\mathbf{R}$ and $\bar{\mathbf{x}} - \bar{\mathbf{r}}$, the goal position $\mathbf{g}_i$ for each particle $i \in \mathcal{P}$ can be computed as

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

To apply forces such as spring damping, a goal velocity $\hat{\mathbf{v}}_i$ for each particle $i \in \mathcal{P}$ can be computed as

$$\hat{\mathbf{v}}_i = \frac{\sum_{j \in \mathcal{P}} m_j \mathbf{v}_j}{\sum_{j \in \mathcal{P}} m_j}. \tag{7}$$

At each timestep, an explicit symplectic Euler integration scheme is used to compute the velocity and position of each particle i as

$$\mathbf{v}_i(t+h) = \mathbf{v}_i(t) + \alpha \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + h\frac{f_{ext}(t)}{m_i}, \tag{8}$$

$$\mathbf{x}_i(t+h) = \mathbf{x}_i(t) + h\mathbf{v}_i(t+h). \tag{9}$$

The second term on the right-hand side of Equation 8 is the Hookean spring force to pull the particle towards goal position $\mathbf{g}_i$. Here, α is the spring stiffness parameter. Müller and colleagues [2005] bound this parameter to the range $0 \leq \alpha \leq 1$. In the presence of no additional external forces and when $\alpha = 1$, the particle is moved directly to the goal position. When $\alpha = 0$, the velocity and position of the particle is not affected by the spring force, and when $\alpha < 1$, the particle moves towards the goal position.

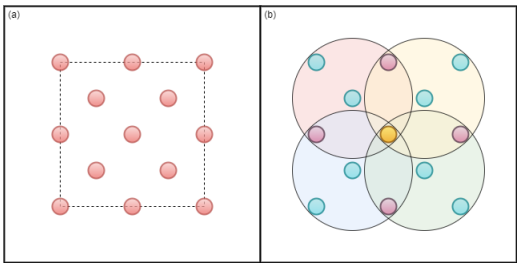

Figure 3: Overlapping Clusters: (a) An object is sampled with a set of particles (in red). (b) The particles are grouped into overlapping clusters (in red, yellow, green, blue). Each particle contributes to the computations for the set of clusters to which it is a member (here, aqua = 1 cluster, purple = 2 clusters, orange = 4 clusters).

3.2 Clustered Shape Matching

A natural extension to shape matching that was also introduced by Müller and colleagues [2005] is clustered shape matching. Clustered shape matching allows for a greater range of motion than basic shape matching because it divides the particles $\mathcal{P}$ into a set of overlapping clusters C, such that the formulations presented in Section 3.1 are applied on a cluster-by-cluster basis. In fact, shape matching is a restricted formulation of clustered shape matching, such that the set of particles $\mathcal{P}$ are all contained in exactly one cluster. Figure 3 graphically depicts a set of overlapping clusters covering an object's particles.

Overlapping clusters can be produced using the fuzzy c-means algorithm discussed by Jones and colleagues [2015]. After randomly selecting a subset of particles as initial seed locations, fuzzy c-means performs the following iterative optimization steps:

(1) Compute spherical cluster membership and *weight of particles in each cluster*, such that a particle *may be assigned to more than one cluster.*
(2) Update each cluster center to be the *weighted* center-of-mass of the particles in that cluster.

When computing a particle's contribution to cluster quantities, we distribute the particle's mass amongst the clusters for which it is a member. This is accomplished by defining a weight $w_{i,c}$ for each particle i in cluster $c \in C$, which indicates how much of particle i's mass contributes to cluster c, and then replacing m_i with $w_{i,c}m_i$ in Equations 1-4 and when computing cluster mass, velocity, and center-of-mass. For instance, the world center-of-mass $\bar{\mathbf{x}}_c$ for a cluster $c \in C$ can be expressed as

$$\bar{\mathbf{x}}_c = \frac{\sum_{i \in \mathcal{P}_c} (m_i w_{i,c})\, \mathbf{x}_i}{\sum_{i \in \mathcal{P}_c} (m_i w_{i,c})}. \tag{10}$$

Once the goal position/velocity of each particle with respect to each cluster is computed, the final goal position/velocity is computed as a weighted average of the goal positions/velocities determined by each cluster for which it is a member; that is,

$$\mathbf{g}_i = \sum_{c \in C} w_{i,c} \mathbf{g}_{i,c}, \tag{11}$$

Algorithm 1: Constructing Multi-resolution Clusters

Data: $\mathcal{P}$ – set of particles
N – number of clusters
r – clustering radius
m – multiplier for radius at each level
Result: H – list containing set of clusters per level

$\ell = 0$
$H = [\,]$
do
  C_ℓ = FUZZY-C-MEANS ($\mathcal{P}$, N, r)
  $H.insert(C_\ell)$
  ℓ += 1
  $N = \max\left\{\lfloor \frac{N}{8} \rfloor, 1\right\}$
  r *= m
while $|C_\ell| > 1$
return H

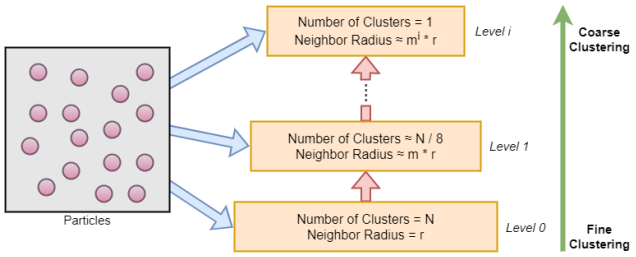

Figure 4: Multi-resolution Clustering.

$$\hat{\mathbf{v}}_i = \sum_{c \in C} w_{i,c} \hat{\mathbf{v}}_{i,c}. \tag{12}$$

Notice that the integration scheme for each particle's velocity (Equation 8) and position (Equation 9) remains the same as that presented in Section 3.1, except that Equation 11 is used as the goal position of the particle.

4 METHODS

We first describe how we perform multi-resolution clustering and then detail how we use it to compute dynamics.

4.1 Multi-resolution Clustering

Our approach performs multi-resolution clustering by iteratively constructing coarser, overlapping clusters on the entire set of particles at each resolution level. More specifically, given a set of particles $\mathcal{P}$, an initial number of clusters N, and a clustering neighbor radius r, we perform fuzzy c-means clustering to generate the finest resolution level of clusters. To construct coarser resolution levels, we reduce the number of clusters N, increase the clustering radius r by a multiplicative factor m, and perform fuzzy c-means on all the particles to generate a coarser set of overlapping clusters. This process is repeated until a single cluster encompasses the entire set of particles. Our multi-resolution clustering approach is presented graphically in Figure 4 and as pseudocode in Algorithm 1.

Algorithm 1 takes the initial number of clusters N and the corresponding cluster neighbor radius r for the finest clustering level (i.e. level 0) as parameters. Good initial values for N and r must be

Algorithm 2: Multi-resolution Timestep Computation

Data: $\mathcal{P}$ – set of particles
H – list containing set of clusters per level
W – list containing weights per level
Result: particle dynamics are updated for current timestep

1 **for** $p \in \mathcal{P}$ **do**
2 $\quad \Delta \mathbf{v}_p = \langle 0.0, 0.0, 0.0 \rangle$
3 **end for**
4 $L = |H| - 1$
5 **for** $\ell \leftarrow L$ *downto* 0 **do**
6 $\quad C_\ell = H[\ell]$
7 $\quad w_\ell = W[\ell]$
8 $\quad$ TIMESTEP($\mathcal{P}$, C_ℓ, w_ℓ)
9 **end for**
10 elapsedTime += dt

Algorithm 3: Weighted Particle Dynamics (TIMESTEP in Algorithm 2)

Data: $\mathcal{P}$ – set of particles
C_ℓ – set of clusters at level ℓ
w_ℓ – weight assigned to level ℓ
Result: weighted particle dynamics

1 **for** $p \in \mathcal{P}$ **do**
2 $\quad \Delta \mathbf{v}_p$ += $w_\ell * \left(dt * gravity + \frac{\alpha}{dt} * (\mathbf{g}_p - \mathbf{x}_p) + k_d * (\hat{\mathbf{v}}_p - \mathbf{v}_p) \right)$
3 $\quad$ **if** $\ell == 0$ **then**
4 $\quad\quad \mathbf{v}_p = \mathbf{v}_p + \Delta \mathbf{v}_p$
5 $\quad\quad \mathbf{x}_p = \mathbf{x}_p + dt * \mathbf{v}_p$
6 $\quad$ **end if**
7 **end for**

selected on a case-by-case basis, but we typically set N to be

$$N = \left\lfloor \frac{|\mathcal{P}|}{n} \right\rfloor, \text{where } 15 \leq n \leq 50. \tag{13}$$

We find that having, at a minimum, roughly 15-50 particles per cluster produces reasonably small clusters while also allowing for sufficient overlap to ensure stability of objects in simulation. The initial value of r is tuned accordingly, based on the value of N, to generate reasonable clustering results.

Furthermore, in line 7 of Algorithm 1, while the number of clusters (N) at each resolution level can be reduced by any arbitrary factor, we choose to divide by 8. Intuitively, this choice is reasonable, as we tend to subdivide a k-dimensional space into 2^k partitions and since we are working in three dimensions; this is similar to an octree [Srihari 1984]. Finally, in line 8 of Algorithm 1, the neighbor radius r is multiplied by a factor m (where $m > 1$) to allow for clustering of the next coarser resolution level. Setting the multiplier m to 2 corresponds to an 8-fold increase in spherical cluster volume, but for some geometries, a slightly larger value improves convergence of the clustering algorithm.

4.2 Applying Multi-resolution Clusters to Dynamics Computations

In our approach, we perform a top-down traversal of the multi-resolution clusters (i.e. from coarsest to finest resolution level) and apply a weighted average of forces accumulated across all the resolution levels. Pseudocode is presented in Algorithm 2.

The TIMESTEP function in Algorithm 2 follows directly from formulations presented in Section 3 for one spatial scale. Within the TIMESTEP function, we compute weighted particle dynamics as defined in Algorithm 3. Here, k_d is a spring damping constant.

Our approach ensures that the actual time integration is handled at the finest level of clusters (i.e. level 0) after the contribution of the coarser resolution levels are accounted for. Furthermore, in our approach, we handle strain limiting [Bargteil and Jones 2014] and collision handling [Jones et al. 2015] operations on the finest set of clusters after each time integration step.

In weighted dynamics, assigning higher weight to any particular resolution level will allow for a greater contribution of the forces accumulated at that level, thereby influencing the behavior of the particles to more closely resemble the behavior that occurs when only that level is used. We present four intuitive weighting functions: 1) uniform, 2) linear, 3) Gaussian falloff, and 4) polynomial profile.

The simplest uniform weighting scheme assigns equal weight to each resolution level; that is, each level ℓ has weight

$$w_\ell = 1. \tag{14}$$

The linear weighting scheme assigns a weight to each resolution level using a linear function. For a scheme that assigns greater weight to coarser levels, each resolution level ℓ has weight

$$w_\ell = \ell + \epsilon. \tag{15}$$

Notice that ϵ is a positive constant to ensure that none of the resolution levels have zero weight. To assign greater weight to finer levels (with a total of L levels), we use

$$w_\ell = (L - \ell - 1) + \epsilon. \tag{16}$$

The Gaussian falloff weighting scheme assigns a weight to each resolution level according to a Gaussian distribution with a mean of 0 and standard deviation of 1 [Kayri and Zirhlioglu 2009]. To assign greater weight to finer levels, we use

$$w_\ell = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\ell^2}. \tag{17}$$

Similarly, to assign greater weight to coarser levels (with a total of L levels), we use

$$w_\ell = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(L-\ell-1)^2}. \tag{18}$$

Finally, for our polynomial profile weighting scheme, we take inspiration from the polynomial form used by Kavan and colleagues [2011] with cloth meshes. To assign greater weight to coarser levels (with a total of L levels), we use

$$w_\ell = \left(1 + b\left(\frac{\ell}{L}\right)\right)^c. \tag{19}$$

Similarly, to assign greater weight to finer levels, we use

$$w_\ell = \left(1 + b\left(\frac{L - \ell - 1}{L}\right)\right)^c. \tag{20}$$

After applying any of the four weighting schemes, we normalize each weight in order to ensure that the sum of weights across all resolution levels is 1; that is, for each level ℓ out of a total of L levels, the weight w_ℓ is normalized to

$$w_\ell = \frac{w_\ell}{\sum_{i=0}^{L-1} w_i}. \tag{21}$$

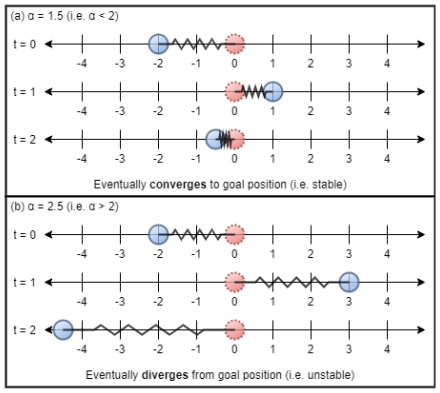

Figure 5: Object Stability When Changing α: (a) With a fixed goal position (in red) and $\alpha < 2$, the particle's position eventually converges towards the goal position. (b) With $\alpha > 2$ the particle's position diverges from the goal position and becomes unstable.

Notice that this normalization allows for the total spring stiffness α and spring damping constants k_d to be distributed across all resolution levels, thereby ensuring stability even with the use of multi-resolution clustering.

Our approach provides additional flexibility in defining weights for each resolution level by enabling the user to manually set weights as opposed to using the predefined weighting functions. We ensure that the manually entered weights are also normalized to sum to 1 by applying Equation 21.

We would also like to explicitly state an observation regarding the range of values for the spring stiffness α which, despite being used to generate results in prior research [Falkenstein et al. 2017; Jones et al. 2015, 2016], has not been formally addressed. More specifically, Müller and colleagues [2005] bound α to the range $0 \leq \alpha \leq 1$. However, by also allowing $1 < \alpha < 2$, the time integration computations in Equations 8-9 enable each particle's position to be "overshot" past the computed goal position while still maintaining object stability. Furthermore, results in prior research [Jones et al. 2015] seem to indicate that setting $1 < \alpha < 2$ may allow for more expressive behavior. Figure 5 presents a simple, one-dimensional example to show how allowing $\alpha < 2$ still enables for stable behavior.

5 RESULTS

In this section, we discuss results produced by applying our multi-resolution clustered shape matching approach, and highlight its effectiveness/versatility with respect to the traditional/baseline approach (i.e. clustered shape matching at a single spatial scale). While we present screenshots from our various animations and discuss the changes observed by applying our approach, these results are most apparent in the supplemental video. Our animations are computed at 30Hz and rendered either as individual particles (colored by nearest cluster) or as a smoothed mesh over the set of particles. Details regarding specific parameter and weight configurations can be found in the supplemental document.

We note that our results use Hookean springs for the underlying dynamics and are not directly comparable to position-based dynamics approaches such as the work of Chentanez and colleagues [2016]. The video results could also be improved by using high resolution render meshes and linear blend skinning; collisions could also be improved by using these meshes at much greater runtime cost. We also note that damping parameters were intentionally set rather low to highlight the dynamics, resulting in some examples appearing "jiggly"; practical examples would likely use more damping.

5.1 Intuition for Selecting Reasonable Animation Parameters

Our multi-resolution clustered shape matching approach allows for many animation parameters to be tuned in order to generate reasonable results. Table 1 summarizes these parameters, and provides intuition into how the animation will be affected by changing the values of these parameters. It is important to note that there is no universal parameter configuration that will work well for all animations. In fact, the definition of a "reasonable" parameter configuration is subjective, and depends on the desired behavior for the animation. Nevertheless, the intuition in Table 1 allows for the artist to construct reasonable estimates for the parameters, which can then be finely tuned to generate the desired behavior.

5.2 Examples

Falling Ball. Our first didactic example demonstrates that our multi-resolution approach allows for unique behavior not captured by the baseline approach. Here, a ball is dropped vertically onto a ground plane (see Figure 6). The ball is sampled with roughly 5K particles, and our approach computes three resolution levels (with 101, 12, and 1 cluster(s), respectively). We first consider three baseline animations, which apply the clustering and dynamics parameters of exactly one level of our multi-resolution approach. With 101 clusters, the ball degenerates into a puddle upon impact. With 12 clusters, the ball is slightly more stiff, although its behavior is quite unpredictable, in that the ball does not return back to its original shape and assume a rest position. With 1 cluster, the ball slightly bounces on impact, but retains its spherical shape throughout the animation.

With our multi-resolution approach, we consider six weighting schemes: 1) polynomial favoring coarser levels [Eq. 19], 2) Gaussian favoring coarser levels [Eq. 18], 3) uniform [Eq. 14], 4) linear favoring coarser levels [Eq. 15], 5) manual weighting favoring middle levels, and 6) Gaussian favoring finer levels [Eq. 17]. We find that with these weighting schemes, the ball squashes in a puddle-like manner upon impact (like 101-cluster baseline), while converging towards a stable, ball-like shape (like 1-cluster baseline and unlike 12-cluster baseline). We also find that varying the weighting function used (and therefore the underlying distribution of weights) allows for the behavior of the ball to be intuitively altered. For instance, the manual weighting scheme demonstrates added fluidity in the ball's behavior when compared to using a Gaussian (favoring coarse levels) scheme.

Hovering Cube. Our second didactic example further demonstrates the robustness of our approach. Here, a cube floating in zero-gravity is stretched by a factor of 2 in the x-direction and

Table 1: Tunable Parameters and Expected Behavior

Type	Parameter	Description and Expected Behavior
Clustering	N	◦ Number of overlapping clusters over particles ◦ *Valid Values*: $N > 0$ ◦ As $N \uparrow$, object deformation richness $\uparrow$ ◦ As $N \downarrow$, object stiffness $\uparrow$ ◦ An initial value for N at the finest resolution level with our approach is $\lfloor \frac{\lvert \mathcal{P} \rvert}{n} \rfloor$, where $15 \leq n \leq 50$
	r	◦ Radius of spherical clusters ◦ *Valid Values*: $r > 0$ ◦ As $r \uparrow$, clusters are larger and may poorly fit object geometry, but FUZZY-C-MEANS is likelier to converge ◦ As $r \downarrow$, clusters are smaller and may better fit object geometry, but FUZZY-C-MEANS may not initially converge
	m	◦ Cluster radius multiplier ◦ *Valid Values*: $m > 1$ ◦ Tuning m influences the cluster radius at each resolution level, which is bounded by intuition for r
Elasticity	α	◦ Hookean spring stiffness pulling particles towards goal positions ◦ *Valid Values*: $0 \leq \alpha \leq 2$ ◦ If $\alpha = 0$, no Hookean force is applied ◦ If $0 < \alpha < 1$, particles are pulled towards goal position ◦ If $\alpha = 1$, particles move directly to goal position (i.e. rigid-body dynamics) ◦ If $1 < \alpha \leq 2$, particles' positions "overshoot" past goal position, but may allow for more expressive behavior
	k_d	◦ Spring damping force constant ◦ *Valid Values*: $0 \leq k_d \leq 1$ ◦ As $k_d \uparrow$, particles' velocities approach the weighted average of velocities for the clusters in which the particles are a member, and objects tend to be less "jiggly"
	$gravity$	◦ Gravitational force constant ◦ *Valid Values*: $gravity \geq 0$ ◦ As $gravity \uparrow$, the gravitational force on each particle (and therefore object) $\uparrow$
Weights	w_ℓ	◦ Contribution of forces at resolution level ℓ ◦ *Valid Values*: $w_\ell \geq 0$ ◦ As $w_\ell \uparrow$ relative to weights for other levels, object behavior is increasingly similar to applying baseline approach with clustering parameters at level ℓ

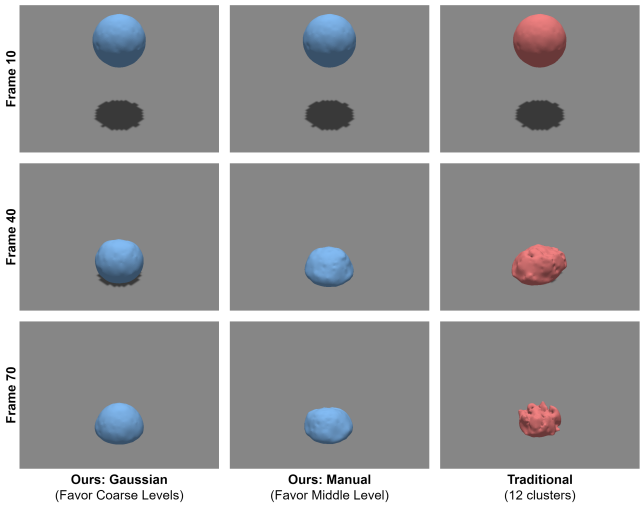

Figure 6: Frames from Falling Ball Animation.

released (see Figure 7). The cube is sampled with roughly 1.3K particles, and our approach computes three resolution levels (with 66, 8, and 1 cluster(s), respectively). Like the previous example,

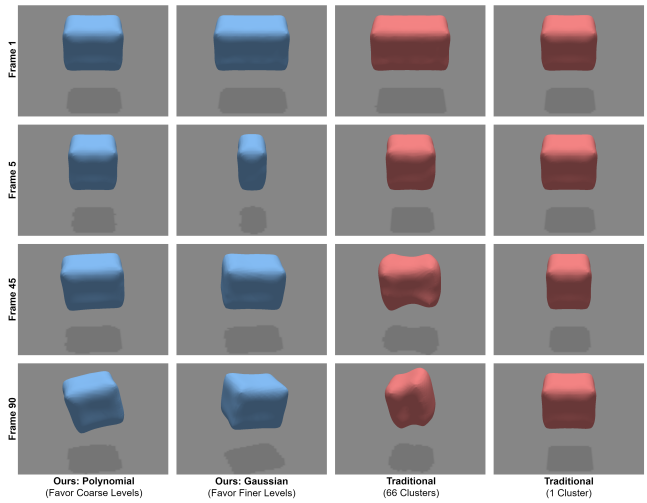

Figure 7: Frames from Hovering Cube Animation.

we first consider three baseline animations. With 66 clusters, the cube demonstrates fluid motion, especially around the corners, and some rotational movement. With 8 clusters, the cube demonstrates some fluid motion around the corners, but otherwise is stiff with no rotational movement. With 1 cluster, the cube demonstrates very rigid motion and no rotational movement.

With our multi-resolution approach, we first consider three weighting schemes: 1) polynomial favoring finer levels [Eq. 20], 2) linear favoring finer levels [Eq. 16], and 3) Gaussian favoring finer levels [Eq. 17]. We find that with all three weighting schemes, the cube demonstrates fluidity in motion, although the behavior is slightly damped (with respect to 66-cluster baseline). We also find that the cubes demonstrate different amounts of rotational movement, which is in part due to the lack of rotational movement at coarser resolution levels. We also consider three additional weighting schemes: 1) uniform [Eq. 14], 2) polynomial favoring coarser levels [Eq. 19], and 3) linear favoring coarser levels [Eq. 15]. We find that with these weighting schemes, the cube demonstrates even more rigid motion (like 1-cluster and 8-cluster baselines), while additionally offering some level of rotational movement (like 66-cluster baseline).

Falling Bunny. Our next example uses more complex geometry to demonstrate the changes in animation behavior produced by our multi-resolution approach when tuning parameters of our predefined weighting functions. Here, a bunny is dropped vertically into a cavity defined by three planes (see Figure 8). The bunny is sampled with roughly 16.5K particles, and our approach computes four resolution levels (with 330, 41, 5, and 1 cluster(s), respectively). We begin by considering two baseline animations as a basis for the range of possible motions. With 1 cluster, the bunny bounces slightly upon impact with the ground plane, but retains its stiff shape. With 330 clusters, the bunny becomes puddle-like on impact and demonstrates rather unstable behavior.

With our multi-resolution approach, we first consider changing the exponent parameter c in the polynomial (favoring finer levels) weighting scheme [Eq. 20] while keeping b fixed at 10. We find that

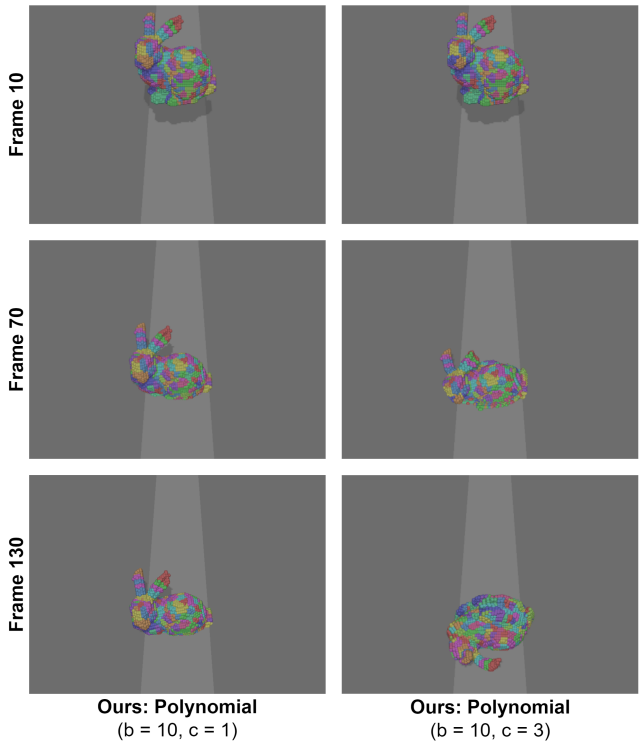

Figure 8: Frames from Falling Bunny Animation.

as the value of c increases from 0 (i.e. uniform weighting), the bunny demonstrates increasingly fluid-like behavior upon impact while remaining stable (unlike 330-cluster baseline). Also, by increasing c (and therefore weight towards finer levels), the bunny demonstrates increasing levels of rotational movement (like 330-cluster baseline). We also consider changing the parameter ϵ when using the linear (favoring finer levels) weighting scheme [Eq. 16]. We find that by increasing ϵ from 0.01 to 1 to 100, the overall stiffness and rigidity of the bunny upon impact is improved. In fact, notice that as ϵ becomes large, the behavior of the bunny approaches that when a uniform weighting scheme is used.

Ball Colliding Into Wall. Our fourth example considers throwing a ball at a wall in zero-gravity (see Figure 9). The ball and wall are each sampled with 1K particles, and our approach computes three resolution levels per object (with 66, 8, and 1 cluster(s), respectively). We begin by considering three baseline animations. Due to collision proxies being poorly defined when a small number of large clusters is used with the baseline approach, with 1 cluster per object, the ball is far away from the wall when it collides with the collision proxies and bounces off, while with 8 clusters per object, the ball comes close to the wall, although it does not interact with the wall particles during collision. With 66 clusters per object, the collision proxies are well-defined and the ball interacts with the wall upon collision, but the ball jitters in a fluid-like motion as it moves away from the wall.

With our approach, since we handle collisions at the finest level of clusters, we can use the same collision proxies as the 66-cluster

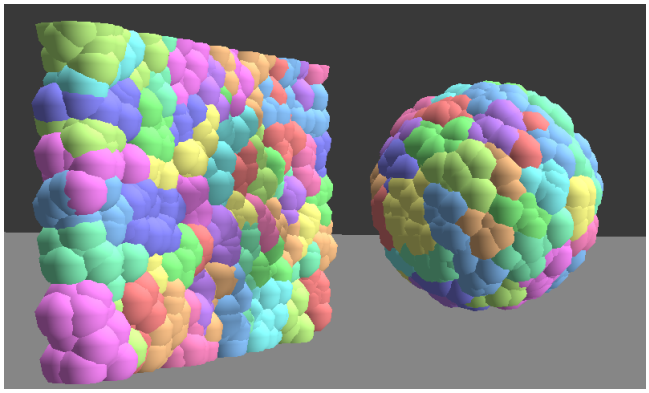

Figure 9: Frame from Ball Colliding Into Wall Animation.

baseline while assigning greater weight towards the coarser resolution levels for the ball so that it remains rigid upon impact (Config. #1). We also manually adjust the weights for the wall to favor the middle (Config. #2) and coarsest (Config. #3) levels, respectively, and find that with a weighting scheme that favors coarser resolution levels, the wall appears slightly more stiff, gets pushed by the ball upon impact, and somewhat dampens the speed at which the ball bounces away from it after impact. Additionally, we demonstrate that by increasing the spring damping k_d from 0.1 to 0.3 to 0.5 for Config. #1, we can reduce jittering and produce more realistic results; we reiterate that some of our examples are intentionally jittery to better highlight differences in animation behavior.

Three Bunnies Collide. In this example, three bunnies collide in zero-gravity, with the left and right bunnies initially moving towards a stationary middle bunny (see Figure 10). Each of these bunnies is sampled with roughly 4.1K particles, and our approach computes four resolution levels per object (with 206, 25, 3, and 1 cluster(s), respectively). With the baseline animation of 206 clusters per object, we find that upon and after impact, each of the bunnies demonstrate some jittering behavior, such as around the ears. With our approach, we first use Gaussian weighting [Eq. 17] and spring stiffness $\alpha = 1.5$ for the left/right bunnies, and polynomial weighting [Eq. 19] and $\alpha = 0.75$ for the middle bunny (Config. #1). We also consider a uniform weighting [Eq. 14] scheme with $\alpha = 1.5$ for all the bunnies (Config. #2). We find that the primary difference with respect to the different configurations is that the direction and fluidity of motion for each bunny upon impact changes due to the differences in the material properties caused by varying the parameters per object.

Plinko. In our next example, a ball falls vertically while interacting with cylindrical rods, similar to a plinko board (see Figure 11). The ball is sampled with roughly 5K particles, and our approach computes three resolution levels (with 101, 12, and 1 cluster(s), respectively). We begin by considering three baseline animations. With 1 cluster, the ball remains stiff and retains its shape upon impact with the ground plane, but instead of bouncing between the rods as desired, it travels directly through them. With 12 clusters, the ball more fluidly bounces between the rods towards the left side of the screen, but it demonstrates constant vibrating movement

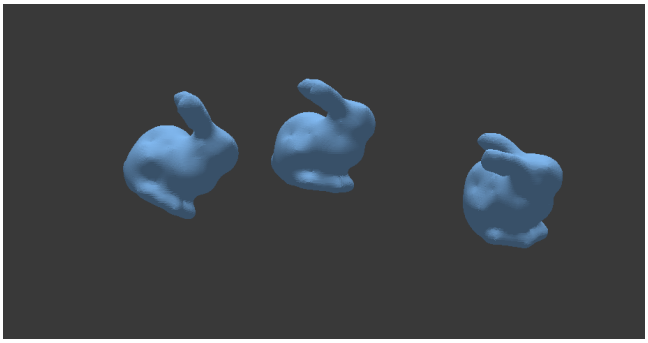

Figure 10: Frame from Three Bunnies Collide Animation.

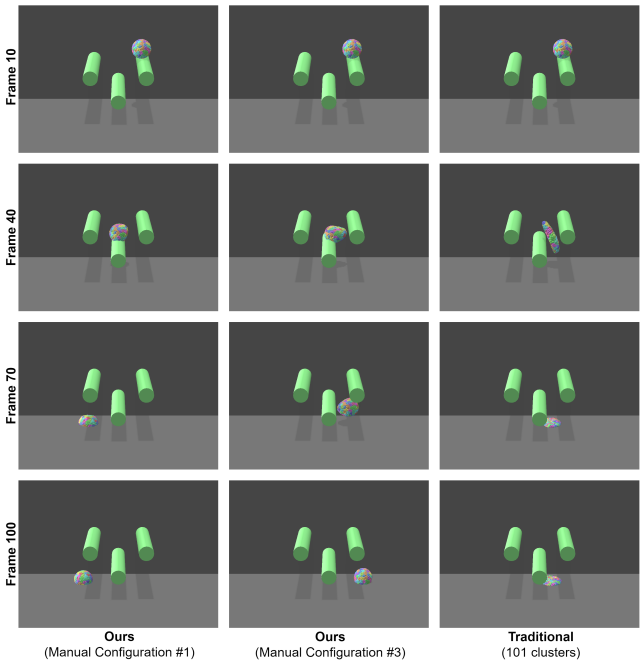

Figure 11: Frames from Plinko Animation.

upon impact with the ground plane. With 101 clusters, the ball is unable to retain its shape as it bounces between the rods towards the right side of the screen, eventually degenerating into a puddle upon impact with the ground.

With our approach, we manually adjust the weights for the ball to favor the coarsest (Config. #1), middle (Config. #2), and finest (Config. #3) levels, respectively. We find that with all three weighting schemes, the ball is able to bounce between the rods while retaining its relative shape, and converges to a rest position upon impact with the ground plane, unlike any of the baseline animations. As such, our approach allows for desired behavior that is not apparent when applying the baseline approach.

Armadillo. In our final example, an armadillo falls vertically onto a pyramid of cylinders (see Figure 1). The armadillo is sampled with roughly 125K particles and our approach computes six resolution levels (with 8192, 1024, 128, 16, 2, and 1 cluster(s), respectively). This is our *largest* example, using a particle set roughly five times larger than that used in recent clustered shape matching research [Jones et al. 2016]. Like previous examples, we begin by considering six baseline animations. As the number of clusters increases, the armadillo's behavior becomes increasingly dynamic/fluid. With 1 and 2 clusters, the armadillo remains stiff (apart from some jittering behavior around the leg) as it collides with the cylinders and falls onto its back. With 16 and 128 clusters, the armadillo falls onto its face while retaining its original shape. With both 1024 and 8192 clusters, the armadillo behaves fluid-like upon impact; with 8192 clusters, it also degenerates into a puddle.

With our approach, we first apply a linear weighting scheme that favors finer resolution levels [Eq. 16]. We find that the armadillo moves towards the left side of the screen and performs a somersault. We also apply a manual weighting scheme that highly favors the finest resolution level, slightly favors the coarsest levels, and assigns minimal weight to the remaining levels. We find that the armadillo eventually falls backwards onto its back before rolling into a position with its legs up in the air and its arms wide across the ground. In both cases, by performing a weighted contribution of the clustering levels, we demonstrate behavior that is not apparent in the baseline approach. Due to the larger number of particles in this example, the timing results for the baseline animations did not correlate with the number of clusters; with a large set of small clusters, particles belonged to fewer clusters, resulting in greater sparsity.

5.3 Timing

Table 2 presents the dynamics timing details for the animations conducted in this paper, all of which were conducted on a machine with 8GB RAM and a 2.50GHz Intel i7-6500U CPU. We note that our

Table 2: Dynamics Timing Details

Example	Animation	Timing (ms per frame)
Falling Ball (5,098 particles)	Baseline: 1 cluster	~3.95
	Baseline: 12 clusters	~7.93
	Baseline: 101 clusters	~9.50
	Ours (average)	***~10.82***
Hovering Cube (1,331 particles)	Baseline: 1 cluster	~0.24
	Baseline: 8 clusters	~0.45
	Baseline: 66 clusters	~0.99
	Ours (average)	***~1.27***
Falling Bunny (16,510 particles)	Baseline: 1 cluster	~22.38
	Baseline: 330 clusters	~89.92
	Ours (average)	***~108.92***
Ball Colliding Into Wall (1,000 particles per object)	Baseline: 1 cluster per object	~0.64
	Baseline: 8 clusters per object	~2.93
	Baseline: 66 clusters per object	~7.59
	Ours (average)	***~8.68***
Three Bunnies Collide (4,131 particles per object)	Baseline: 206 clusters per object	~58.83
	Ours (average)	***~63.90***
Plinko (5,098 particles)	Baseline: 1 cluster	~5.15
	Baseline: 12 clusters	~8.56
	Baseline: 101 clusters	~10.75
	Ours (average)	***~12.20***
Armadillo (124,631 particles)	Baseline: 1 cluster	~235.36
	Baseline: 2 clusters	~386.97
	Baseline: 16 clusters	~1254.37
	Baseline: 128 clusters	~1718.34
	Baseline: 1024 clusters	~1456.65
	Baseline: 8192 clusters	~1067.08
	Ours (average)	***~1818.50***

codebase is designed for research, not performance, and runs on a single CPU, but that the overhead of our approach is minimal. From the timing data, we find that our approach takes longer to compute dynamics per frame compared to the baseline approach, which essentially computes a single scale. However, our approach takes less time than the sum of computing each spatial scale separately, likely due to the fact that we apply strain limiting only at the finest set of clusters. Nevertheless, we believe this additional cost is justified given the enhanced behavior enabled by our multi-resolution clustering approach.

6 CONCLUSIONS

A limitation of our multi-resolution approach is that the dynamics computations for each timestep take longer than that when applying the traditional approach. However, due to the method in which we compute particle dynamics, our approach can be parallelized by simultaneously computing the contribution of each resolution level. Furthermore, our code is limited to a sequential process on a CPU, but it may be possible to map our framework to a GPU, similar to what has already been done for shape matching and position-based dynamics (PBD) [Chentanez et al. 2016; Fratarcangeli and Pellacini 2013]. As such, considering methods of parallelizing our clustered shape matching code may be an interesting avenue for future work. We also found that our different strategies for automatically setting the stiffness at various levels yielded similar results, any of which should be acceptable in practice. We did not consider changing damping parameters between resolution levels, but this would be straightforward.

One fundamental limitation of the clustered shape matching approach is the bounds on the stiffness α. We initially hoped our multi-resolution approach would support very stiff materials, but because α is bounded by 2, such materials are not feasible unless the timestep is reduced. The "jiggliness" of our objects could be reduced by increased damping, but we chose to keep damping low to moderate because with larger damping values, the influence of cluster size becomes more difficult to demonstrate.

A recurring limitation noted in prior work is a lack of theoretical underpinnings for clustered shape matching. More specifically, due to the lack of mathematical tools to analyze the clustered shape matching approach, Jones and colleagues [2016] state that there is no analytical understanding as to how the method behaves when the number of particles, number of clusters, cluster radius, and/or timestep is varied. This concern carries forward to our approach, which applies clusters at multiple spatial scales. However, as Jones and colleagues [2016] also note, the finite element method was used for several decades before a mathematical framework was developed to analyze its properties. As such, this limitation is in no way a hindrance to the practicality and versatility of the clustered shape matching framework.

As shown in this paper, we have expanded the clustered shape matching framework by introducing a multi-resolution clustering approach for enhancing the elastic behavior of objects in animation. We showed that objects animated using our approach demonstrate an increased stiffness that is not visible when using only a large set of small clusters in the traditional approach, as well as relatively richer deformations that are not apparent when using only a small set of large clusters. We have also allowed for greater flexibility in tuning animation behavior by providing the ability to specify properties such as spring stiffness and weights on a per-object basis.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable feedback.

REFERENCES

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.

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.

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.

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

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 the Tenth International Conference on Motion in Games* (Barcelona, Spain) *(MIG '17)*. ACM, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/3136457.3136473

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.

Marco Fratarcangeli and Fabio Pellacini. 2013. A GPU-Based Implementation of Position Based Dynamics for Interactive Deformable Bodies. *Journal of Graphics Tools* 17, 3 (2013), 59–66. https://doi.org/10.1080/2165347X.2015.1030525 arXiv:https://doi.org/10.1080/2165347X.2015.1030525

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.

Liang He, Ricardo Ortiz, Andinet Enquobahrie, and Dinesh Manocha. 2015. Interactive Continuous Collision Detection for Topology Changing Models Using Dynamic Clustering. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 47–54.

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 the ACM SIGGRAPH Conference on Motion in Games* (Paris, France).

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*.

Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. 2011. Physics-Inspired Upsampling for Cloth Simulation in Games. *ACM Trans. Graph.* 30, 4, Article 93 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1964988

Murat Kayri and G Zirhlioglu. 2009. Kernel smoothing function and choosing bandwidth for non-parametric regression methods. *Ozean Journal of Applied Sciences* 2, 1 (2009), 49–54.

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.

Chao Ma, Jia-Bin Huang, X.K. Yang, and Ming-Hsuan Yang. 2015. Hierarchical convolutional features for visual tracking. *Proc. Int. Conf. Mach. Learn.* (01 2015), 3074–3082.

Matthias Müller. 2008. Hierarchical Position Based Dynamics. In *VRIPHYS*.

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

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. https://doi.org/10.1145/1073204.1073216

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).

SN Srihari. 1984. Multiresolution 3-d image processing and graphics. *Multiresolution Image Processing and Analysis* (1984), 224–236.

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* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 87–94.