Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016)
Ladislav Kavan and Chris Wojtan (Editors)

Real-time Simulation of Large Elasto-Plastic Deformation with Shape Matching

Nuttapong Chentanez[1,2] Matthias Müller[1] Miles Macklin[1]

[1]NVIDIA
[2]Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University

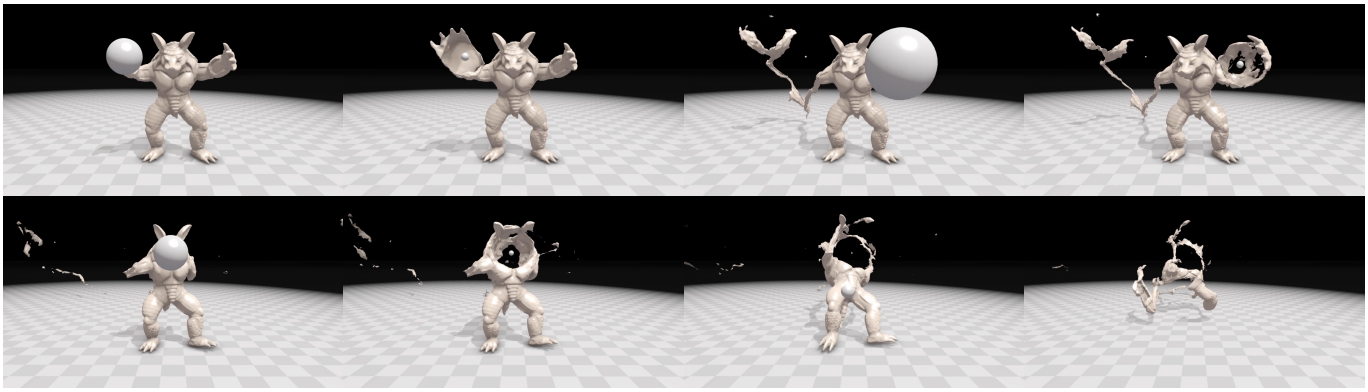

Figure 1: *Snapshot from a real-time simulation of an armadillo shaped solid being shot in various places,* $\varepsilon_\gamma = 0.1$, $\upsilon = 0.9$, $\kappa = 0$ *and* $K = 0.1$.

Abstract

Shape matching is a popular method for simulating deformable objects in real time as it is fast and stable at large time steps. Although shape matching can simulate large elastic deformation and ductile fracturing, until now, they are limited to scenarios with relatively small plastic deformation. In this work, we present a method for simulating deformable solids undergoing large plastic deformation and topological changes using shape matching within the position based dynamics (PBD) framework. This expands the versatility of PBD which was previously shown to be able to simulate rigid bodies, liquids, gases, cloth, and deformable solids with moderate plastic deformation. Our novel contributions include local particle re-sampling, cluster re-sampling and skinning of an explicitly tracked surface mesh.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Animation/Physical Simulation—Physical Simulation

1. Introduction

Shape matching is a popular method for simulating deformable solids represented by particles. It is based on a compliant constraint formulation which is stable and efficient enough for real-time applications. Because it is a geometric method it is well suited to elastic, or near rigid, materials where there is a well defined goal configuration. While simulating elastic materials has been well explored, the added ability to simulate large plastic deformation is necessary to simulate many real-world materials such as taffy, dough, and foams.

A variety of extensions to shape-matching to support plastic deformation have been proposed [MHT05], [Cho14] and [JML*16]. These methods typically adjust the reference configuration of particles in response to deformation using a heuristic constitutive model. This approach, however, is not suitable for large plastic deformation where particle separation can cause severe under-sampling, and the original shape-matching groups can no longer represent all modes of deformation.

Particle re-sampling strategies such as the ones proposed in [SB12], [APKG07] [ATT12], and [JWJ*14] exist, but are not immediately

DOI: 10.2312/sca.20161233

applicable to the re-sampling of shape matching constraints. In this work we propose a method that locally inserts and deletes shape-matching clusters based on simple deformation criteria, allowing essentially unbounded deformation. Our method is compatible with any shape-matching-based simulator.

We also address the problem of rendering the resulting simulation. The output of shape matching algorithm is a set of affine transformations that are often used to drive linear blend skinning [BM82], [MTLT88] of an attached visual mesh. This is a simple and efficient method that works well for materials where deformation is small, and topological changes are not present. Our goal, however, is to support large scale deformations and allow materials to tear and fracture. In this work we use explicit mesh tracking [CM15] to update the visual mesh topology, and describe a new skinning algorithm that respects the changes to the underlying simulation.

Our main contributions are:

1. A local resampling scheme for particles and shape matching clusters.
2. An improved skinning method for the explicitly tracked surface mesh.

2. Related Work

Position based dynamics (PBD) [MHR06], which formulates constrained dynamics directly in terms of positions, has become a popular method for real-time simulation. It has been extended to multi-resolution representations [Mül08], the simulation of elastic rods [USS14], fluids [MM13], smoke, and granular materials [MMCK14]. A comprehensive survey on other recent developments of PBD can be found in [BMM15].

Müller et al. [MHT05] introduced shape matching for simulating deformable solids. The best fit transformations of the current particles configuration are used to define goal positions that particles are pulled towards. The method was later optimized with fast summation by Rivers and James [RJ07] to improve computation speed when the particles lie on a lattice. Later, Steinemann et al. [SOG08] extended the fast summation approach to handle adaptive lattices. To improve the stability of shape matching when used with an explicit integrator Bargteil and Jones [BJ14] proposed to apply strain limiting. Choi [Cho14] devised a strain measure in the context of shape matching and use it for simulating plastic deformation and ductile fracture. To give artists additional control over material behavior Jones et al. [JML*15] proposed the use of clusters with weights. They also devised a plasticity and fracture model for shape matching [JML*16] inspired by the finite element simulation of [BWHT07]. While extensions to the basic shape matching method exist for the simulation of fracture, mesh cutting and moderate plastic deformations, handling very large plastic deformations in the context of shape matching has not been addressed so far.

Solids undergoing plastic deformation have also been simulated with other methods. The finite element method is among the post popular methods used in computer graphics. O'Brien et al. [OBH02] used FEM for the simulation of ductile fracture. Later, Bargteil et al. [BWHT07] extended the constitutive model to handle large plastic flow. Wojtan et al. [WT08] addressed the problem of handling thin features often occuring under large plastic deformations. They also proposed a method to handle topological changes, an additional problem that arises when objects undergo large plastic deformations [WTGT09]. Instead of augmenting Lagrangian models to handle plastic flow, Goktekin et al. [GBO04] extended the Eulerian fluid simulator approach to handle viscoelastic solids which was later extended to the case of multiple interacting liquids [LSSF06]. Stomakhin et al. [SSC*13] found that the material point method (MPM) which alternates between a Lagrangian and an Eulerian representation is well suited for the simulation of snow with its unique and distinctive behaviour including melting and solidifying [SSJ*14]. Ram et al. [RGJ*15] proposed an alternative MPM formulation to handle plastic flow in foams and sponges.

Particle based models are particularly well suited for the simulation of large plastic deformations because handling thin features, topological changes and volume conservation is straight forward. Müller et al. [MKN*04] formulated a continuum-based model on particles by computing the deformation gradient from the particle locations with Moving Least Square (MLS) and were able to simulate solids undergoing plastic deformation. This approach was later extended to handle ductile fracture in [PKA*05]. Kaiser et al. [KAG*05] developed a unified approach to handle the transition from solid to fluid. Instead of using a continuum based method, Clavet et al. [CBP05] modeled viscoelastic fluids with a mass-spring system where springs are inserted and removed dynamically. The smoothed particles hydrodynamics method (SPH), mostly used for liquids, has also been extended to handle plastic solids. Solenthaler et al. [SSP07] proposed using SPH for the unified modeling fluids, deformable solids, and rigid bodies. To reduce the computation time considerably, Peer et al. [PICT15] devised an implicit viscosity term for SPH that requires only a single projection step. Hieber and Koumoutsakos [HK08] proposed a Lagrangian particle method for simulating elasto-plastic solids that do not store a rest configuration, but instead perform a least square fit to the deformation gradient at each step. To robustly derive the deformation gradient from points Gerszewski et al. [GBB09] combine affine transformations that best approximate local motions. The method does not store the rest configuration and hence can only handle limited elastic deformations. Jones et al. [JWJ*14] proposes the use of an embedded space, in addition to the rest space and the world space for simulating elasto-plastic solids. A common problem of particle based approaches is that the deformation gradient is not unique in sparse regions which yields numerical instabilities. A popular method to circumvent this problem is to additionally store orientation information on the particles [MKB*10], [MC11] and [FGBP11].

There is a large body of work on particle re-sampling. Here we only list work that is closely related to our approach. Schechter and Bridson [SB12] adapted the Fast Poisson Disk Sampling [Bri07] method for evenly sampling a fluid domain with fluid and ghost particles. Similarly, Ando et al. [ATT12] re-sample FLIP [ZB05] particles in thin fluid regions to better preserve thin sheets. In their hybrid model, Chentanez et al. [CM15] used Fast Poisson Disk sampling for generating particles when the grid representation of the liquid is converted to particles. Instead of complete re-sampling Adams et al. [APKG07] proposed to split and merge existing particles to speed up SPH - based fluid simulation. Jones et al. [JWJ*14]

perform particle splits and merges to improve stability of their point-based solid simulation. Our particle re-sampling strategy utilizes the Fast Poisson Disk Sampling idea.

For rendering, we skin a surface to the particles. This area has been researched exensively too. Of the most popular appoaches are linear blend skinning [BM82], [MTLT88], multi-weight enveloping [WP02], animation space [MMG06], and dual quaternion [KCvO08] skinning. An excellent resource containing the state-of-the art methods can be found in the course note by Jacobson et al. [JDKL14].

3. The Method

We represent objects by a set of particles and an explicit surface mesh. Our simulation method comprises three main steps. At every time step the states of the particles are first advanced by simulating elastic and plastic deformation. After this, the explicit surface is advected with the particles via skinning and topological changes are handled if necessary. Finally, the object is re-sampled with particles and the clusters are updated. We will now describe these steps in more detail.

3.1. Simulating Elastic and Plastic Deformation

We use clustered shape matching [MHT05] as a constraint in a parallel position based dynamics constraint solver [MMCK14] and handle plastic deformation with the method proposed by Choi [Cho14]. Other plastic deformation models such as that of Jones et al. [JML*16] could be used as well.

More specifically, we discretize solid objects using a set of particles, $p_i \in P$, with masses, m_i, positions, p_i and velocities v_i. To advance the simulation by a time step Δt, we first apply external forces to the velocities via $v_i \leftarrow v_i + \Delta t f_{ext}(x_i)$ and use these updated velocities to compute predicted positions $x_i^* \leftarrow x_i + \Delta t v_i$. The constraint solver then performs a number of projection iterations on the predicted positions. In each iteration all positional constraints are projected in parallel. The correction vectors from different constraints are accumulated in a variable Δx_i for each particle. These variables are used to update $x_i^* \leftarrow x_i^* + \Delta x_i / n_i$, where n_i is the number of constraints affecting particle i. After a given number of solver iterations the velocities are updated via $v_i \leftarrow \frac{1}{\Delta t}(x_i^* - x_i)$ and $x_i \leftarrow x_i^*$. For more details on each step and for contact handling, see [MMCK14].

We use clustered shape matching to hold the particles together. Clusters are overlapping sub-sets of the particles. Let C be the set of particles in a cluster, F_P the plastic deformation matrix, which is initialized to identity, and let q_i be the local position of particle i with respect to the cluster's center of mass which we denote by c. Following the shape matching approach, we first compute

$$A_{pq} = (\sum_{i \in C} m_i (p_i - c) q_i^T) F_P^T, \tag{1}$$

$$A_{qq} = F_P (\sum_{i \in C} m_i q_i q_i^T) F_P^T, \tag{2}$$

and use polar decomposition $A_{pq} A_{qq}^{-1} = R S_{pqqq}$ to extract the rotational part of the matched linear transformation from the rest to the current pose. The goal position of particle i is then computed as $g_i = R F_P q_i + c$. We update $\Delta x_i \leftarrow \Delta x_i + K(g_i - x_i)$, where $0 \leq K \leq 1$ is the stiffness of the cluster.

To update F_P, we use the plastic deformation model by Choi [Cho14]. First, we diagonalize A_{qq} to get

$$A_{qq} = V_{qq} \mathrm{diag}(\lambda_1^{qq}, \lambda_2^{qq}, \lambda_3^{qq}) V_{qq}^T. \tag{3}$$

$$\tag{4}$$

We then compute the optimal stretch matrix S using:

$$S = V_{qq} [(V_{qq}^T S_{pq} V_{qq}) \circ \bar{\Lambda}^{qq}] V_{qq}^T, \tag{5}$$

where $\bar{\Lambda}_{ij}^{qq} = 2/(\lambda_i^{qq} + \lambda_j^{qq})$ and $\circ$ is the Hadamard product which produces the element-wise multiplcation of its operands. Next, we decompose $S = V \Lambda V^T$, where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$. We then use the logarithmic strain, $\varepsilon_l = V \log(\Lambda) V^T$ for our strain measure and use a multiplicative plasticity model as in [ITF04]. The plastic deformation F_P can then be updated as

$$F_P \leftarrow V (|\Lambda|^{-\frac{1}{3}} \Lambda)^{\gamma} V^T F_P, \tag{6}$$

where $\gamma = \min(\frac{\upsilon(||\varepsilon_l|| - \varepsilon_\gamma)}{||\varepsilon_l||}, 1)$ indicates how much of the elastic deformation is transferred to the plastic component, υ is the plastic flow rate, and $||\varepsilon_l|| = \max_i(|\log(\lambda_i)|)$. If work hardening is desired, the plastic yield, ε_γ, is updated by $\varepsilon_\gamma \leftarrow \kappa ||\varepsilon_l|| \Delta t$, where κ is a coefficient controling the amount of plastic hardening.

To improve stability, in the case of violent interaction, we clamp the diagonal entries of the diagonal matrix $(|\Lambda|^{-\frac{1}{3}} \Lambda)^{\gamma}$ to be between 0.8 to 1.2, which roughly prevents F_p from changing more than 20% in a single time step.

3.2. Visual Mesh Skinning

We skin the visual mesh to the moving particles using a two-step process. First, we interpolate the orientation, R, of each shape matching cluster to particles. This is done by using the normalized average of the quaternions of the clusters influencing each particle, which is the best fit rotation in a geodesic distance measure [Gra01]. Let the normalized average quarternion of particle i be Q_i. Each visual mesh vertex stores particle indices l_a, local positions z_a and weights w_a, where $1 <= a <= 4$. The position of the visual mesh vertex, y, is then computed as $y = \sum_{a=1}^{4} w_a (Q_{l_a}^M z_a + x_{l_a})$, where $Q_{l_a}^M$ denotes the rotation matrix corresponding to the quaternion Q_{l_a}. We do not directly skin the visual mesh to the clusters because particles contain positional detail that clusters cannot represent. Since particles do not store orientation needed for skinning we interpolate it from the clusters.

After the new vertex positions are determined, we resolve topological changes and ensure good triangle quality with the method proposed by [CMMK15]. During this step, we delete visual mesh vertices for which there is no particle within a distance of $2r$. This can potentially cause a topological split.

3.3. Re-Sampling

The re-sampling step comprises a series of sub-steps. First, invalid particles are removed and new particles seeded in under-sampled

regions within the visual mesh. Second, the current clusters are updated, invalid clusters are removed and new clusters added if necessary. Finally, the skinning weights of the visual mesh vertices are adjusted and the mesh updated. We now describe the details of each step.

3.3.1. Remove Invalid Particles

As the simulation proceeds, some particles may get too close to others or may move far outside the visual mesh. These particles no longer contribute significantly to the simulation. Hence, we flag particles for deletion when one or both of the following conditions are true: (a) A particle's index is smaller than the index of its closest neighbor and the distance to the closest neighbor is less than ηr, where r is the particle radius and η is a small constant. We use $\eta = 0.1$ in all our examples. (b) A particle is outside the inflated visual mesh and is not referenced by any vertex of the visual mesh for skinning. The inflated visual mesh has the same topology as the visual mesh but with vertices displaced along their normals by a small amount. We use $0.25r$ in all our examples. For mesh inside-outside tests we use ray casting.

3.3.2. Seed New Particles

Due to plastic deformation during the simulation, particles may move far apart and create under-sampled regions. To avoid this, we perform local re-sampling by attempting to seed T particles around each particle belonging to a significantly deformed cluster. We use $T = 5$ for all examples. A cluster is considered significantly deformed if the Frobenius norm of its plastic deformation matrix F_P exceeds a threshold (1.8 in our examples). The candidate positions are generated by uniformly sampling the space between two concentric spheres of radius r and $2r$ [Bri07]. If a candidate position is farther than r from its nearest particle, lies within the inflated visual mesh and is not inside other scene geometry, we seed a particle there with the same physical quantities as the source particle.

3.3.3. Update Clusters

As particles are deleted and added, we need to update clusters to reflect these changes. For each cluster C we remove referenced deleted particles and add all newly created particles x_i for which $||x_i - c|| < r^C$, where c the cluster's center of mass and r^C its undeformed radius. The local position, q_i, of an added particle i is computed as $q_i = W(x_i - c)$, where W is the transformation matrix from world space to the plastically deformed local space of the cluster. W is computed as $W = F_P^{-1} A_{qq} A_{pq}^{-1}$.

If a cluster changes during this step, we need to shift the local positions of the particles so that the local center of mass is at the origin. We first compute the center of mass using the rest positions of the particles $\bar{c} = \frac{\sum_{i \in C} m_i q_i}{\sum_{i \in C} m_i}$. Then we update the local positions by $q_i \leftarrow q_i - \bar{c}$.

3.3.4. Remove Invalid Clusters

During the simulation, clusters may have a lot of particles added or removed, may undergo large plastic deformation or may overlap significantly with other clusters. When this happens the clusters may no longer faithfully capture the underlying deformation field and need to be re-sampled. We achieve this by simply removing such clusters. When conditions are appropriate, the removal will trigger the creation of new clusters to be added and hence we achieve re-sampling.

A cluster is removed if

- it has fewer than half the initial number of particles or
- it has more particles than twice the initial number of particles or
- the Frobeinius norm of its plastic deformation matrix F_P exceeds a threshold (2.0 in our examples) or
- all of its particles are members of more than $M_{\max}$ other clusters.

3.3.5. Add New Clusters

When clusters are removed, their member particles will have fewer clusters influencing them and newly created particles are not influenced by any cluster. In this case we have to add new clusters. First we create a list L_{cand} of the particles referenced by less than $M_{\min}$ clusters. We then select a random particle from L_{cand}, append its index to a list, L_{cen}, and mark all particles in L_{cand} within radius r^G from it, where $r^G \leq r^C$ is the radius for creating clusters. We repeat this selection step until all the particles in L_{cand} are marked.

We then create one cluster for each particle in L_{cen} containing all the particles within distance r^C. The plastic deformation matrix, F_P, of the new cluster is set to identity. We embed the plastic deformation information directly into the local positions of the member particles, by setting $q_i \leftarrow W_{\text{new}}(x_i - c)$, where W_{new} is interpolated from old clusters, including the ones that got deleted in this time step.

We use a method similar to the one proposed by Bargteil et al. [BWHT07] for interpolating W_{new}. First, we compute $U = W^{-1} = A_{pq} A_{qq}^{-1} F_P^{-1}$ of all the old clusters and $G = \frac{U^T U - I}{2}$. After this we interpolate G_{new} by using a weighted average of G of the old clusters, where the weights are proportional to the overlapping volume of the spheres with radius r^C centering at the clusters. We then diagonalize $2G_{\text{new}} + I = V_{\text{new}} D_{\text{new}} V_{\text{new}}^T$, where $D_{\text{new}} = \text{diag}(\lambda_1^{\text{new}}, \lambda_2^{\text{new}}, \lambda_3^{\text{new}})$ and V_{new} is a rotation matrix. For added stability, we clamp $\lambda_1^{\text{new}}, \lambda_2^{\text{new}}, \lambda_3^{\text{new}}$ to be between 0.25 and 4. We then compute $W_{\text{new}} = V_{\text{new}}^T \text{diag}(\frac{1}{\sqrt{\lambda_1^{\text{new}}}}, \frac{1}{\sqrt{\lambda_2^{\text{new}}}}, \frac{1}{\sqrt{\lambda_3^{\text{new}}}}) V_{\text{new}}$. The plastic yield ε_γ is also interpolated using the same weights.

The cluster generation described above is executed once per time step. Therefore, it potentially could take up to $M_{\min}$ time steps for a particle get included in $M_{\min}$ clusters. In practice however, newly generate particles are always close to some existing clusters and get included into them right away.

3.3.6. Update visual mesh and the skinning weights as needed

When particles get deleted or change their clusters membership, the skinning indices, skinning weights, and skinning local positions of the visual vertices need to be updated. First, particle orientations, Q_i, have to be interpolated from clusters again because they are no longer valid if particles change their cluster membership. Then for each visual vertex, we check if its skinning indices, l_a, refer to particles that got deleted. If so, we let l_a reference the four closest particles and set the skinning weights, w_a, in proportion to their

inverse distance. These skinning weights are normalized to sum to 1. We re-compute the skinning locations $z_a = Q_{l_a}^{-1}(y - x_{l_a})$ only when needed, namely, when l_a changes or when Q_{l_a} changes due to re-sampling.

In some examples, we also utilize a volume conservation strategy similar to Thuerey et al. [TWGT10] which moves visual mesh vertices along their normal by an amount proportion to the difference between a target volume and the current volume divided by the current surface area. This is done independently for each connected component of the visual mesh. While this step could be repeated until the volume differs from the target volume by no more than ε, we found it sufficient in our situation to only do it once per time step. If a visual vertex is moved by this process, we need to re-compute its local skinning position, z_a, as above. As the movement tends to be very small, we found that re-computing skinning indices and weights is not necessary in this case.

3.3.7. Handling multiple connected components

During simulation, an object could be split into several components due to tearing, user manipulation or other mechanisms. Unless particles move far apart very quickly in a single time step, the method described so far tends to prevent splitting because clusters tend to form between particles that should belong to different components.

We address this problem by first identifying the connected components of the visual mesh. Each vertex of the visual mesh stores the ID of the connected component it belongs to. We use the skinning indices to propagate these IDs to the particles. If a particle is referenced by vertices from different components, we create an identical particle with a small positional offset for each component. The ID information is then propagated inward to the particles far away from surface in a breadth-first search fashion. After this we check for each cluster whether all referenced particles belong to the same component. If this is not the case we simply mark the cluster for deletion. The steps above are done before invalid particles are removed.

The particle removal step is modified to only consider the distance to the closest particle within the same component ID. A newly created particle in the particle seeding step gets its ID from the source particle. The cluster update step is modified so that only particles with the same component ID are included. The cluster creation step now only creates new clusters from particles with the same component ID, and interpolates information from the old clusters that have the same component ID. Finally, when we compute the skinning indices of a visual vertex, we consider only the particle with the same component ID as the vertex.

When different components of the visual mesh merge, they get the same component ID and the particles from previously separated components will now be able to form a cluster. When a user cut is performed, we simply cut the visual mesh into different components by deleting all triangles that cross the cut plane and then mark the vertices involved to not contribute to pairing score during the entire pairing step of [CMMK15].

4. Results and Discussion

We wrote a GPU implemented our algorithm. Our un-optimized prototype runs in real-time or at interactive rates on a single NVIDIA GTX TitanX in all examples including simulation, surface tracking, re-sampling and rendering. We use $r = 0.1$, $r^G = 3r$, and $r^C = 4r$ in all examples, where our objects have size between $20r$ to $100r$. The animations can be found in one of our accompanying videos. Snapshots of an animation of the armadillo being shot in various locations are shown in Figure 1. Figure 2 shows an armadillo shaped solid being dropped on the ground with various flow rates and plastic yield thresholds yielding a variety of behaviors. Figure 3 demonstrates different behaviors for various hardening coefficients. Figure 4 shows snapshots of a bunny shaped solid being pulled and shot.

Snapshots of an animation of a cow shaped mesh being caught between moving bars is shown in Figure 5. A taffy pulling machine is simulated as shown in Figure 6. These examples demonstrate that our re-sampling method holds up under very large plastic flow.

We also simulate dough being squeezed, rolled over and split as shown in Figure 7. The dough is manipulated by hands in real-time via a hand tracking device Figure 8.

Without our resampling step, the simulation will either become unstable, if F_s is not clamped, or not be able to undergo large deformation, if F_s is clamped. The simulation also would not be able to undergo topological change. These situations are demonstrated in one of our accompanying videos.

The timings and statistics for the examples are shown in Table 1. The surface mesh topological change and triangle mesh quality improvement usually dominate the running time, as we use a relatively high resolution surface mesh compared to the particle density. Our re-sampling method takes a significant amount of time in the particle deletion and addition steps due to ray casting for inside / outside tests whose execution time depends strongly on the resolution of the surface mesh. The ray casting function alone is responsible for about 60% of the re-sampling running time. Nonetheless, the overall frame rate is still real-time or interactive. Depending on how large the plastic deformation is, the percentage of the particles that we consider seeding new particles nearby range from 15% on average for the bunny to 80% on average for the cow cutting example. We use the volume conservation strategy in the bunny, the taffy pulling machine and the dough examples.

5. Conclusion and Discussion

We presented the first shape-matching-based solid simulation that can handle very large plastic deformation through the use of particle and cluster re-sampling. Our method affects only the shape matching constraints in the context of a unified particle physics simulation [MMCK14]. This allows the solid to interact with liquids, gases, rigid bodies and other types of particles in the framework.

We do not attempt to conserve the total mass of the objects in this work. We argue that conserving the volume of the visual mesh, which is what the end-user sees, is sufficient in many practical use

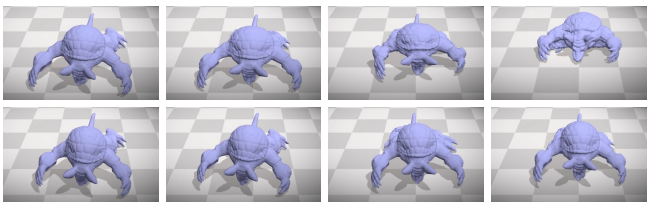

Figure 2: *Final frame of a real-time simulation of an armadillo shaped solid dropped onto the ground with various flow rates and plastic yield thresholds. The top row simulations use* $\varepsilon_\gamma = 10^{-4}$*, while the bottom row simulations use* $\varepsilon_\gamma = 10^{-2}$*. From the leftmost column to the rightmost column,* υ *used are* $0.1, 0.2, 0.5$*, and* 0.9 *respectively. All examples use* $\kappa = 0$*,* $K = 0.9$*.*

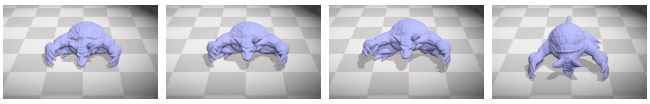

Figure 3: *Final frame of a real-time simulation of an armadillo shaped solid dropped onto the ground with various work hardening coefficients. From the leftmost column to the rightmost column, the* κ *used are* $0, 0.1, 1$*, and* 5*. All examples use initial* $\varepsilon_\gamma = 10^{-4}$*,* $\upsilon = 0.9$*, and* $K = 0.9$*.*

cases. Moreover, as we keep good particle distributions during simulations, the total mass is approximately conserved anyway.

To treat multiple objects, different component IDs can be assigned to each of them. If the topological change and ray-casting steps are carried out for each of them independently, the different objects never merge. We do not explicitly handle cluster splitting due to fracture in our implementation, but it can be added using either methods by Choi [Cho14] or Jones et al. [JML*16].

While our method is described within the context of shape matching constraints and the PBD framework, we believe it can be readily adapted to pure shape matching simulation used in Bargteil and Jones [BJ14], Jones et al. [JML*15], Jones et al. [JML*16].

Ray casting to determine if a point is inside the surface mesh is by far the most significant bottleneck of our re-sampling method. We actually do not need the inside-outside information to be 100% precise. A method that determines the inside-outside information only approximately could be employed to improve the running time of our re-sampling greatly.

Currently, our method may create clusters with particles belonging to the same connected component of the visual mesh within radius r^C.

Our method of working with component IDs does not prevent a complicated single component from merging if it folds over itself within a radius of r^C. We do not find this to be problematic in our examples. This is because particles that are about to merge tend to have their relative velocity approaching each other anyway and usually cause the visual mesh to merge. To prevent this type of merge one could use the geodesic distance along overlapping particles instead of a Euclidean distance measure to decide whether particles are to be included in a cluster. We view this as an interesting future work.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (July 2007). 1, 2

[ATT12] ANDO R., THUREY N., TSURUNO R.: Preserving fluid sheets with adaptively sampled anisotropic particles. *Visualization and Computer Graphics, IEEE Transactions on 18*, 8 (Aug 2012), 1202–1214. 1, 2

[BJ14] BARGTEIL A. W., JONES B.: Strain limiting for clustered shape matching. In *Proceedings of the Seventh International Conference on Motion in Games* (New York, NY, USA, 2014), MIG '14, ACM, pp. 177–179. 2, 6

[BM82] BADLER R., MORRIS M.: Modelling flexible articulated objects. In *Computer Graphics' 82* (1982). 2, 3

[BMM15] BENDER J., MÜLLER M., MACKLIN M.: Position-based simulation methods in computer graphics. *EUROGRAPHICS Tutorial Notes, Zürich, May 4-8* (2015). 2

[Bri07] BRIDSON R.: Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 2, 4

[BWHT07] BARGTEIL A. W., WOJTAN C., HODGINS J. K., TURK G.: A finite element method for animating large viscoplastic flow. *ACM Transactions on Graphics 26*, 3 (July 2007), 16:1–16:8. 2, 4

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation* (2005), 219–228. 2

[Cho14] CHOI M. G.: Real-time simulation of ductile fracture with oriented particles. *Computer Animation and Virtual Worlds 25*, 3-4 (2014), 455–463. 1, 2, 3, 6

[CM15] CHENTANEZ N., MULLER M.: Coupling 3d eulerian, height-field and particle methods for interactive simulation of large scale liquid phenomena. *Visualization and Computer Graphics, IEEE Transactions on 21*, 10 (Oct 2015), 1116–1128. 2

[CMMK15] CHENTANEZ N., MÜLLER M., MACKLIN M., KIM T.-Y.: Fast grid-free surface tracking. *ACM Trans. Graph. 34*, 4 (July 2015), 148:1–148:11. 3, 5

[FGBP11] FAURE F., GILLES B., BOUSQUET G., PAI D. K.: Sparse Meshless Models of Complex Deformable Solids. *ACM Transactions on Graphics 30*, 4 (July 2011), Article No. 73. 2

[GBB09] GERSZEWSKI D., BHATTACHARYA H., BARGTEIL A. W.: A point-based method for animating elastoplastic solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 133–138. 2

[GBO04] GOKTEKIN T. G., BARGTEIL A. W., O'BRIEN J. F.: A method for animating viscoelastic fluids. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 463–467. 2

[Gra01] GRAMKOW C.: On averaging rotations. *Int. J. Comput. Vision 42*, 1-2 (Apr. 2001), 7–16. 3

[HK08] HIEBER S. E., KOUMOUTSAKOS P.: A lagrangian particle method for the simulation of linear and nonlinear elastic models of soft tissue. *Journal of Computational Physics 227*, 21 (2008), 9195 – 9215. Special Issue Celebrating Tony LeonardâĂŹs 70th Birthday. 2

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation* (2004), pp. 131–140. 3

[JDKL14] JACOBSON A., DENG Z., KAVAN L., LEWIS J.: Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses* (2014). 3

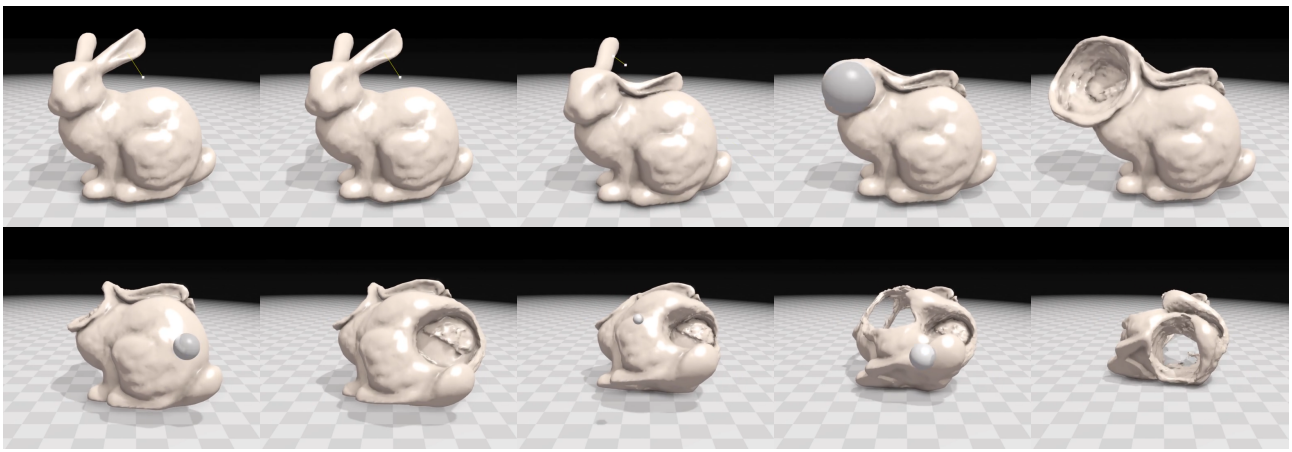

Figure 4: *Snapshot from an a real-time simulation of a bunny shaped solid being pulled and shot in various places.* $\varepsilon_\gamma = 0.1$, $\upsilon = 0.9$, $\kappa = 0$ *and* $K = 0.1$.

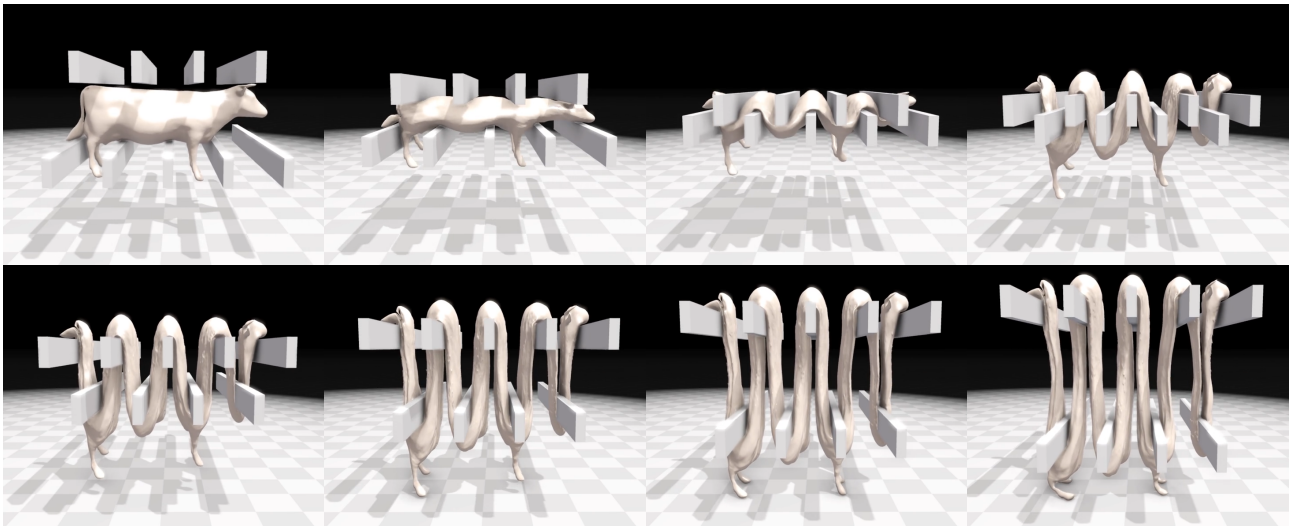

Figure 5: *Snapshot from an animation of a cow shaped solid being caught between several moving rigid bars.* $\varepsilon_\gamma = 10^{-4}$, $\upsilon = 0.99$, $\kappa = 0$ *and* $K = 0.5$.

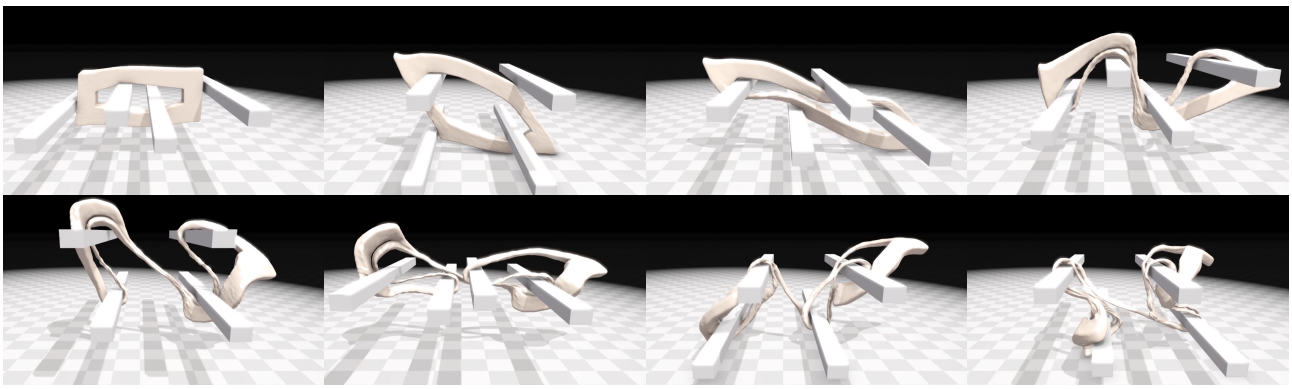

Figure 6: *Snapshot from an animation of a taffy pulling machine simulation.* $\varepsilon_\gamma = 10^{-4}$, $\upsilon = 0.99$, $\kappa = 0$ *and* $K = 0.99$.

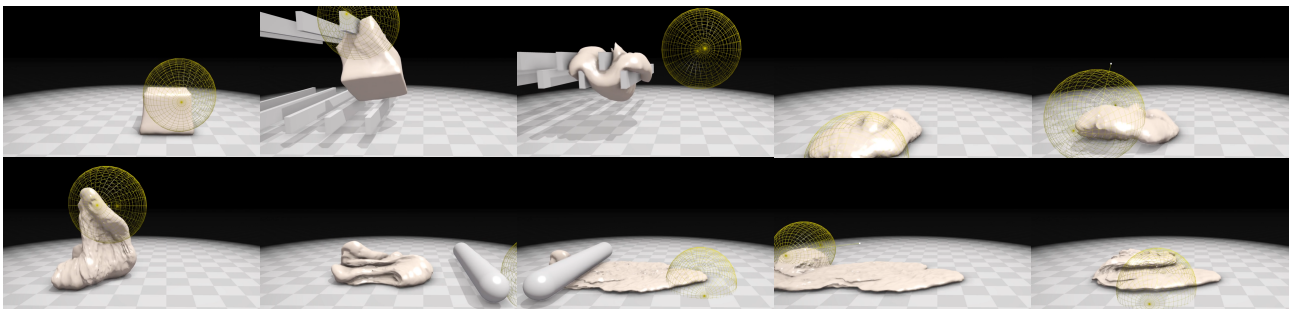

Figure 7: *Snapshot from a real-time simulation of dough being squeezed, rolled, and folded over.* $\varepsilon_\gamma = 0.1$, $\upsilon = 0.9$, $\kappa = 0$ *and* $K = 0.1$.

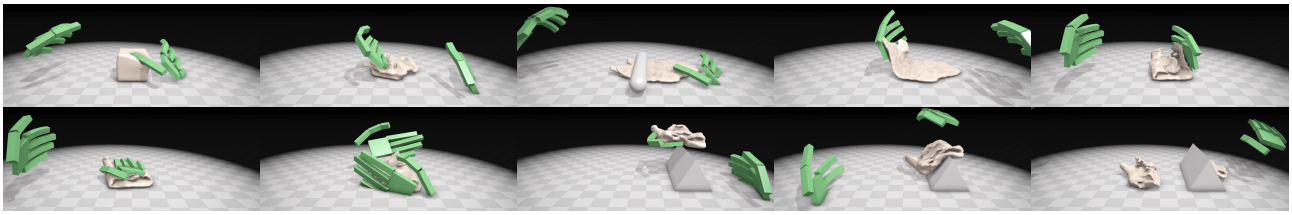

Figure 8: *Snapshot from a real-time simulation of dough being manipulated by hands via a hand tracking device.* $\varepsilon_\gamma = 0.1$, $\upsilon = 0.9$, $\kappa = 0$ *and* $K = 0.5$.

	Sim	Topo	Resampling							Total	Par	%Seed	Clus	Par/Clus	Tris	MaxE
			BH	PD	PA	CS	SK	Oth	RTotal							
Armadillo	2.61	12.37	0.20	2.77	5.78	0.76	1.09	0.32	10.92	25.90	11707	38.7	697	71	60k	0.017
Bunny	4.77	18.87	0.15	3.15	3.99	0.88	1.37	0.40	9.94	33.58	37883	16.5	1387	127	36k	0.017
Cow	1.76	26.05	1.16	1.87	6.81	0.51	1.09	0.37	11.81	39.62	4452	80.0	394	56	239k	0.04
Taffy	2.33	34.99	1.63	0.77	7.79	0.69	3.67	0.47	15.02	52.34	6188	65.5	512	95	342k	0.04
Dough	2.12	7.38	0.19	0.57	4.32	0.54	1.41	0.30	7.33	16.83	7073	50.1	335	108	46k	0.01

Table 1: *Average timings in millisecond and statistics of the examples. Each row corresponds to the average over several runs similar to the situations shown in the video. Sim is the simulation time, Topo is the surface mesh topological change and improvement time, Resampling is our resampling method's time, Total is total time for simulation+surface tracking+resampling, Par is the number of particles, %Seed is the percentage of particles being considered for seeding additional particles nearby, Clus is the number of clusters, Par/Clus is the average number of particles per cluster, Tris is the number of triangles of the surface mesh and MaxE is the maximum edge length for surface tracking (the minimum edge is one tenth of the maximum edge length). BH is the building hash table for ray-casting time, PD is the particle deletion time, PA is the particle addition time, CS is the cluster resampling time(update,delete,add), SK is the skinning time (compute indices, weight and local position updates), Oth is for the time it takes for the remaining of the resampling steps and RTotal is the total time for the resampling step.*

[JML*15] JONES B., MARTIN A., LEVINE J. A., SHINAR T., BARGTEIL A. W.: Clustering and collision detection for clustered shape matching. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games* (New York, NY, USA, 2015), MIG '15, ACM, pp. 199–204. 2, 6

[JML*16] JONES B., MARTIN A., LEVINE J. A., SHINAR T., BARGTEIL A. W.: Ductile fracture for clustered shape matching. In *Proceedings of the ACM SIGGRAPH symposium on Interactive 3D graphics and games* (Feb 2016). 1, 2, 3, 6

[JWJ*14] JONES B., WARD S., JALLEPALLI A., PERENIA J., BARGTEIL A. W.: Deformation embedding for point-based elastoplastic simulation. *ACM Trans. Graph. 33*, 2 (Apr. 2014), 21:1–21:9. 1, 2

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRÉ P., GROSS M.: A unified lagrangian approach to solid-fluid animation. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics* (Aire-la-Ville, Switzerland, Switzerland, 2005), SPBG'05, Eurographics Association, pp. 125–133. 2

[KCvO08] KAVAN L., COLLINS S., ŽÁRA J., O'SULLIVAN C.: Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph. 27*, 4 (Nov. 2008), 105:1–105:23. 3

[LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. In *the Proceedings of ACM SIGGRAPH 2006* (Aug. 2006), pp. 812–819. 2

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (July 2011), 92:1–92:10. 2

[MHR06] MÜLLER M., HENNIX B. H. M., RATCLIFF J.: Position based dynamics. *Proceedings of Virtual Reality Interactions and Physical Simulations* (2006), 71–80. 2

[MHT05] MÜLLER M., HEIDELBERGER B., TESCHNER M.: Meshless deformations based on shape matching. In *Proc. SIGGRAPH 2005* (2005), pp. 471–478. 1, 2, 3

[MKB*10] MARTIN S., KAUFMANN P., BOTSCH M., GRINSPUN E.,

GROSS M.: Unified simulation of elastic rods, shells, and solids. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 3 (2010), 39:1–39:10. 2

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *the ACM SIGGRAPH 2004 Symposium on Computer Animation* (July 2004), pp. 141–151. 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013), 104:1–104:12. 2

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 2, 3, 5

[MMG06] MERRY B., MARAIS P., GAIN J.: Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25*, 4 (Oct. 2006), 1400–1423. 3

[MTLT88] MAGNENAT-THALMANN N., LAPERRIÈRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88* (Toronto, Ont., Canada, Canada, 1988), Canadian Information Processing Society, pp. 26–33. 2, 3

[Mül08] MÜLLER M.: Hierarchical position based dynamics. *Proceedings of Virtual Reality Interactions and Physical Simulations* (2008). 2

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21*, 3 (July 2002), 291–294. 2

[PICT15] PEER A., IHMSEN M., CORNELIS J., TESCHNER M.: An implicit viscosity formulation for sph fluids. *ACM Trans. Graph. 34*, 4 (July 2015), 114:1–114:10. 2

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (July 2005), 957–964. 2

[RGJ*15] RAM D., GAST T., JIANG C., SCHROEDER C., STOMAKHIN A., TERAN J., KAVEHPOUR P.: A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2015), SCA '15, ACM, pp. 157–163. 2

[RJ07] RIVERS A. R., JAMES D. L.: Fastlsm: Fast lattice shape matching for robust real-time deformation. In *ACM Transactions on Graphics (Proc. SIGGRAPH 2007)* (2007), vol. 26(3), pp. 82:1–82:6. 2

[SB12] SCHECHTER H., BRIDSON R.: Ghost sph for animating water. *ACM Trans. Graph. 31*, 4 (July 2012), 61:1–61:8. 1, 2

[SOG08] STEINEMANN D., OTADUY M. A., GROSS M.: Fast adaptive shape matching deformations. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 87–94. 2

[SSC*13] STOMAKHIN A., SCHROEDER C., CHAI L., TERAN J., SELLE A.: A material point method for snow simulation. *ACM Trans. Graph. 32*, 4 (July 2013), 102:1–102:10. 2

[SSJ*14] STOMAKHIN A., SCHROEDER C., JIANG C., CHAI L., TERAN J., SELLE A.: Augmented mpm for phase-change and varied materials. *ACM Trans. Graph. 33*, 4 (July 2014), 138:1–138:11. 2

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid–solid interactions: Research articles. *Comput. Animat. Virtual Worlds 18*, 1 (Feb. 2007), 69–82. 2

[TWGT10] THUEREY N., WOJTAN C., GROSS M., TURK G.: A Multiscale Approach to Mesh-based Surface Tension Flows. *ACM Transactions on Graphics (SIGGRAPH) 29 (4)* (July 2010), 10. 5

[USS14] UMETANI N., SCHMIDT R., STAM J.: Position-based elastic rods. In *ACM SIGGRAPH 2014 Talks* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 47:1–47:1. 2

[WP02] WANG X. C., PHILLIPS C.: Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 129–138. 3

[WT08] WOJTAN C., TURK G.: Fast viscoelastic behavior with thin features. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 47:1–47:8. 2

[WTGT09] WOJTAN C., THÜREY N., GROSS M., TURK G.: Deforming meshes that split and merge. *ACM Trans. Graph. 28*, 3 (July 2009), 76:1–76:10. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *the Proceedings of ACM SIGGRAPH 2005* (Aug. 2005), pp. 965–972. 2