Strain Limiting for Clustered Shape Matching

Adam W. Bargteil*
University of Utah

Ben Jones†
University of Utah

Abstract

In this paper, we advocate explicit symplectic Euler integration and strain limiting in a shape matching simulation framework. The resulting approach resembles not only previous work on shape matching and strain limiting, but also the recently popular *position-based dynamics*. However, unlike this previous work, our approach reduces to explicit integration under small strains, but remains stable in the presence of non-linearities.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation.

Keywords: Strain Limiting, Shape Matching, Position-based Dynamics

1 Introduction

Almost a decade ago, Müller and colleagues [2005] introduced *shape matching* for physics-based computer animation. In this approach objects are discretized into a set of particles, $p_i \in \mathcal{P}$, with rest positions, $\mathbf{r}_i$, that follow a path, $\mathbf{x}_i(t)$, in world-space through time. At each frame, shape matching solves for the rotation matrix, $\mathbf{R}$, and a translation vector, $\mathbf{x}_{cm} - \mathbf{r}_{cm}$, that minimize

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

The best translations are given by the center-of-mass in the rest and world space, respectively. The rotation, $\mathbf{R}$, is computed through a polar decomposition. Intuitively, this computation yields the least-squares best-fit rigid transformation from the rest pose to the current deformed pose. This transformation allows us to define goal positions, $\mathbf{g}_i$,

$$\mathbf{g}_i = \mathbf{R}\left(\mathbf{r}_i - \mathbf{r}_{cm}\right) + \mathbf{x}_{cm}. \tag{2}$$

Hookean springs are then used to define forces that move the particles toward the goal positions. This basic approach was extended by including linear and quadratic global deformations, cluster-based deformations, and plasticity.

In their seminal work, Müller and colleagues [2005] also used linear stability analysis to describe the criteria under which the explicit, symplectic Euler time integration is guaranteed to be stable in this shape matching framework. Unfortunately, this analysis breaks down in the presence of non-linearities, such as many overlapping shape matching clusters.

More recently, *position-based dynamics* (PBD) [Müller et al. 2007; Stam 2009; Bender et al. 2014] has become extremely popular, especially in real-time and interactive applications. In this paradigm, "forces" act directly on positions rather than indirectly through velocities. The approach is frequently described in the language of constraints. For example, instead of a spring force, two particles will be constrained to be a certain distance apart. Constraints may take a variety of forms [Bender et al. 2014] and different constraints may be in opposition, which naturally leads to the exploration of different constraint solvers [Macklin et al. 2014]. In any case, constraints are rarely solved exactly, which allows for deformations over time. With some care in the formulation of constraints and choice of constraint solver, PBD can be guaranteed to be stable, which makes it extremely appealing for real-time computer animation.

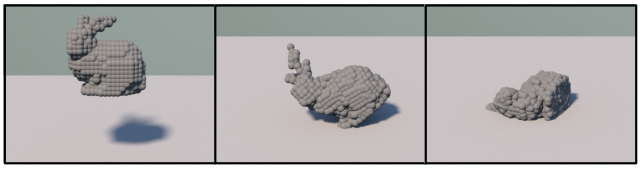

Figure 1: *A bunny simulated with our approach.*

While these two approaches are related, there is a very critical difference—shape matching integrates second-order spring forces through time to move particles toward goal positions; in contrast, PBD updates positions directly, essentially using first-order springs. While very useful for a variety of graphical applications, the direct positional updates violate Newton's first and second laws of motion. This limitation manifests in highly damped deformations, a lack of elastic oscillations, and poor energy and momentum preservation. If treated in this fashion, gravity would produce linear paths not parabolic arcs.

A related approach is *strain limiting*, which was introduced by Provot [1995] almost 20 years ago.[1] Like PBD, strain limiting works through constraints and suffers from the limitations discussed earlier. However, strain limiting typically tries to enforce an upper-bound on strain and is applied after time integration. Thus, the limitations are ameliorated because under small strains, strain limiting has no effect. In contrast, PBD replaces time integration with (typically) zero-strain constraints and achieves deformation by not iterating to convergence.

In this paper, we advocate pairing second-order physics simulation through clustered shape matching with strain limiting. While neither of these ideas are new, we are the first to combine them. The resulting method is extremely practical: it is fast, stable even in the presence of the non-linearities introduced by the overlapping clusters, and preserves the second-order dynamics of clustered shape matching under smaller strains.

2 Methods

Clustering To create the clusters in our cluster-based shape matching we allow the user to specifiy a neighborhood radius, d. Then we iteratively choose a random particle, $p_i \in \mathcal{P}$, and create a new cluster, $c \in \mathcal{C}$, centered at p_i, that includes all particles within distance d of p_i. Formally, the set of particles in the cluster is

$$\mathcal{P}_c = \{p_j \mid \|\mathbf{x}_i - \mathbf{x}_j\| \leq d\} \subseteq \mathcal{P}. \tag{3}$$

Iterations continue until all particles are placed into at least one cluster. The mass of a particle is distributed among its clusters and

*adamb@cs.utah.edu
†benjones@cs.utah.edu

[1] Our formulation very closely resembles that suggested by Bridson and colleagues [2002].

Algorithm 1 Strain Limiting for Shape Matching

1: `compute_goal_positions(0)`
2: $\mathbf{v}_i^* = \mathbf{v}_i(t) + \alpha\frac{\mathbf{g}_i - \mathbf{x}_i(t)}{h} + h\frac{\mathbf{f}_{ext}(t)}{m_i}$
3: $\mathbf{x}_i^0 = \mathbf{x}_i(t) + h\mathbf{v}_i^*$
4: **for** $j = 0$ **to** iters **do**
5: `compute_goal_positions(`γ`)`
6: $\mathbf{x}_i^{j+1} = \omega\mathbf{g}_i + (1-\omega)\mathbf{x}_i^j$
7: **end for**
8: $\mathbf{x}_i(t+h) = \mathbf{x}_i^{\text{iters}}$
9: $\mathbf{v}_i(t+h) = \frac{\mathbf{x}_i(t+h) - \mathbf{x}_i(t)}{h}$

this fact is taken into account when computing the cluster's total mass and center of mass. This weighting ensures that the center of mass of the clusters is the same as the center of mass of the particles and that spring forces obey Newton's third law. Our randomized algorithm is simple and allows user control over the size of shape matching clusters, but it does suffer from significant mass lumping errors that lead to non-uniform distribution of mass, which in turn introduces unintuitive intertial effects and dynamics that do not necessarily preserve symmetries of the underlying particle distributions. For example, an axis aligned scale of a cube can induce rotations. We do not doubt that more sophisticated algorithms could achieve improved results.

Animation Runtime Our approach is summarized in Algorithm 1. The function `compute_goal_positions()`, which computes goal positions by averaging over shape-matching clusters, is summarized in Algorithm 2. Our algorithm has several parameters: the timestep, h, a gain for the internal spring forces, α, a relaxation coefficient for the constraint solver, ω, the number of constraint solver iterations, iters, and the strain limiting threshold, γ. We typically set the timestep to the framerate (60 Hz). $\alpha \in [0, 1]$ is related to the spring stiffness, k, through the substitution $\alpha = h^2k/m$ and is the same value used by Müller and colleagues [Müller et al. 2005]. If $\alpha = 1$ then the positions will move to the goal positions in line 3. Note that this does not guarantee that all constraints are solved. ω is a relaxation constant that we typically set to 1, though Macklin and colleagues [2014] note faster convergence with *over*-relaxation—values between 1 and 2. We typically allow 3-5 iterations of the constraint solver.

Given the strain-limiting threshold, γ, and letting $\mathbf{g}_{ic}$ be the goal position for particle p_i given by shape matching cluster c, we enforce strain-limiting constraints of the form

$$\beta = \frac{\|\mathbf{x}_i - \mathbf{g}_{ic}\|}{w_c} < \gamma, \tag{4}$$

where w_c is the *cluster width*,

$$w_c = \max_{p_i \in \mathcal{P}_C} \|\mathbf{r}_i - \mathbf{r}_{cm}\|, \tag{5}$$

which accounts for the spatial scale of the cluster. In this formulation, $\gamma = 0$ corresponds to equality constraints; we typically use $\gamma \in (0.1, 0.5)$. So that elastic forces attempt to undo all deformation, we pass 0 to `compute_goal_positions()` in line 1 of Algorithm 1.

3 Discussion

The primary difference between our algorithm and PBD is the inclusion of "internal forces" in line 2 of Algorithm 1. Indeed if we

Algorithm 2 `compute_goal_positions(`γ`)`

1: **for all** $p_i \in \mathcal{P}$ **do**
2: $\mathbf{g}_i = 0$
3: $n_i = 0$
4: **end for**
5: **for all** $c \in \mathcal{C}$ **do**
6: Compute $\mathbf{R}$ and $\mathbf{x}_{cm}$
7: **for all** $p_i \in \mathcal{P}_c$ **do**
8: $\mathbf{g}_{ic} = \mathbf{R}(\mathbf{r}_i - \mathbf{r}_{cm}) + \mathbf{x}_{cm}$
9: $\beta = \frac{\|\mathbf{x}_i - \mathbf{g}_{ic}\|}{w_c}$
10: $\mathbf{g}_i \mathrel{+}= \mathbf{g}_{ic} + \min\left(\frac{\gamma}{\beta}, 1\right)(\mathbf{x}_i - \mathbf{g}_{ic})$
11: n_i ++
12: **end for**
13: **end for**
14: **for all** $p_i \in \mathcal{P}$ **do**
15: $\mathbf{g}_i /= n_i$
16: **end for**

set $\alpha = 0$ (and, typically, $\gamma = 0$ as well) our solver reduces to a form of PBD. While this may seem like a small change, it is actually quite significant. Consider a simple mass point connected to a single spring with zero rest length. Our integrator will oscillate around the anchor point. In contrast, when using PBD if the constraint is satisfied for two subsequent timesteps, the velocity is set to zero and all motion ceases. This effect can be seen for a simple example in the accompanying video. The importance of second-order effects can also be seen in Figure 2, where PBD results in energy and momentum loss compared to shape matching when simulating a simple rotating cube.

On the other hand, setting iters $= 0$ (or $\omega = 0$) is equivalent to the original shape matching integrator [Müller et al. 2005]. This integrator works well when forces are linear in positions, but many overlapping shape matching clusters introduce non-linearities that can cause this integrator to become unstable. By employing strain limiting, we ensure that forces, and consequently velocities, remain bounded and the simulation remains stable.

As a simple first example, we take a cube and stretch it by 50% in the x-direction. We show the result for a variety of parameter settings. For a large neighborhood radius, we get a single cluster. With a smaller radius we generate 29 clusters, which results in more interesting and local behavior. In our next set of examples, instead of stretching the cube we drop it on the ground. Our final example is of the bunny bouncing on the ground. This example has 2020 particles grouped into 26 clusters. Note that most of our examples intentionally do not include damping forces in order to highlight the advantages of our approach. We include one additional example of the bunny with damped springs, which results in more natural behavior.

Regarding computation time, our current implementation is dominated by the `compute_goal_positions()` routine. Our approach adds an additional call to this routine compared to PBD and several calls compared to traditional shape matching. However, our program still generally runs at real-time rates as a sequential process on a CPU. For example, for each 60 Hz frame, the damped bunny example took 0.370 ms for shape matching (everything in Algorithm 1 except lines 4-7) and 0.713 ms for strain limiting (lines 4-7 in Algorithm 1).

UU IR Author Manuscript

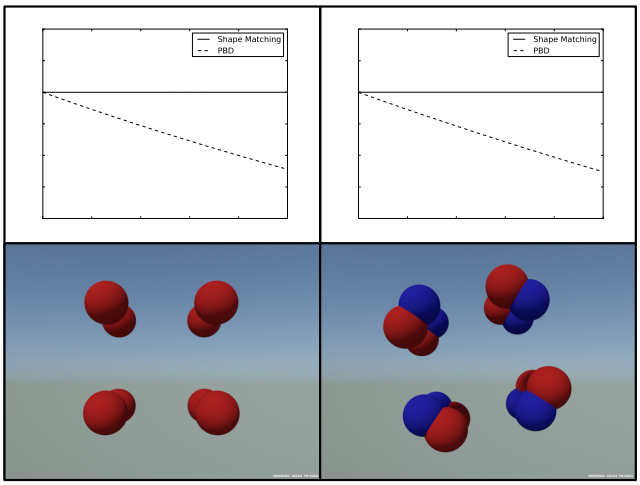

Figure 2: *Shape matching preserves kinetic energy and angular momentum of a cube rotating counter-clockwise, while PBD does not. The red spheres are from a shape matching simulation; the blue spheres are simulated with PBD.*

Limitations and Future Work Our approach to creating shape-matching clusters is simple and effective, however we believe better results could be achieved with more sophisticated algorithms that, for example, introduce multi-scale clusters. We would also like to address the mass lumping errors so that geometric symmetries are preserved in the dynamics. Finally, our current implementation is limited to a sequential process on a CPU. However, we believe our algorithm should parallelize well and, like shape matching and PBD, map very well to a GPU.

Conclusion Given the success of strain limiting in a variety of computer graphics simulations [Provot 1995; Bridson et al. 2002; Thomaszewski et al. 2009; Wang et al. 2010], it is not surprising that it works well in the shape matching context. We expect that the improved results afforded by our approach will lead to its speedy adoption.

Acknowledgments

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

References

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

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3, 594–603.

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

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

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

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, 147–154.

STAM, J. 2009. Nucleus: Towards a unified dynamics solver for computer graphics. In *International Conference on Computer-Aided Design and Computer Graphics*, 1–11.

THOMASZEWSKI, B., PABST, S., AND STRASSER, W. 2009. Continuum-based strain limiting. *Comput. Graph. Forum 28*, 2, 569–576.

WANG, H., O'BRIEN, J., AND RAMAMOORTHI, R. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graph. 29*, 6, 156:1–156:10.