DEFORMATION EMBEDDING FOR POINT-BASED ELASTOPLASTIC SIMULATION

by

Stephen J. Ward

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computing

School of Computing

The University of Utah

August 2012

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of Stephen J. Ward

has been approved by the following supervisory committee members:

Adam W. Bargteil	, Chair	3/19/2012
		Date Approved
Mike Kirby	, Member	3/19/2012
		Date Approved
Peter Pike Sloan	, Member	3/19/2012
		Date Approved

and by Alan Davis , Chair of

the Department of School of Computing

and by Charles A. Wight, Dean of the Graduate School.

ABSTRACT

We present a straightforward, easy-to-implement, point-based approach for animating elastoplastic materials. The core idea of our approach is the introduction of *embedded space*—the least-squares best fit of the material's rest state into three dimensions. Together with *plastic offsets* that map embedded space to rest space, the embedded space allows us to robustly estimate the deformation gradient, compute elastic forces, and account for plastic flow. We additionally introduce an estimate for the volume of a particle, opening the door for nonuniform sampling, and describe a technique to increase the robustness of point-based elastic simulation. Our approach can handle arbitrarily large elastic deformations and extreme plastic deformations. Because the approach is point-based, there is no need for complex remeshing—the corresponding operation is a simple neighborhood query in embedded space. We demonstrate our approach on a variety of examples that display a wide range of material behaviors.

"You know, for kids!" But especially for Jeana.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Computer animation of elastoplastic materials, such as modeling clay, chewing gum, and bread dough has long been a goal of computer graphics, probably because these materials demonstrate such intriguing behavior. As a field we have made progress toward this goal and elastoplastic materials have recently been showcased in special effects, for example, the honey in *Bee Movie* [18] and the food in *Ratatouille* [9]. However, significant limitations in current techniques remain; some handle only limited plastic deformation, some handle only limited elastic deformation, and others require complex remeshing methodologies.

We present a straightforward, easy-to-implement, point-based approach for animating elastoplastic materials. Our approach can handle arbitrarily large elastic deformations and extreme plastic deformations. Because the approach is point-based, there is no need for complex remeshing—the corresponding operation is a simple neighborhood query.

A material that has undergone plastic deformations does not, in general, have a zero-stress state that can be realized in three-dimensional space. Put another way, the *rest space* is not embeddable in three-dimensional space. This fact poses a challenge because elastic forces depend on a mapping from rest space to deformed or *world space*. Some authors have addressed this challenge by keeping plastic offsets from an initial rest state, but this places limitations on the amount of plastic deformation that is possible. Others have abandoned the rest state and keep elastic offsets, resulting in limited elastic deformations and/or artificial plasticity. Borrowing an idea from Wicke and colleagues [22], we take a compromise approach and store an *embedded space*—the weighted least-squares best embedding of rest space into three dimensions. Combined with per particle *plastic offsets* that map from embedded space to rest space, the embedded space gives us a robust mapping from rest space to world space and allows us to handle arbitrary elastic deformations and extreme plastic deformations.

While our primary contribution is the development of our embedding into three-

dimensions, we also suggest ways of improving robustness for particle-based animation of elastic bodies and an approach for estimating volume that allows for nonuniform samplings. We demonstrate our approach on a variety of examples that display a large range of material behaviors.

CHAPTER 2

RELATED WORK

The pioneering work of Terzopoulos and colleagues [20, 21] first addressed elastoplastic deformations almost 25 years ago. Perhaps foreshadowing their use in *Ratatouille*, elastoplastic materials played a key role in the short film *Cooking with Kurt* [5]. Elastoplastic materials have also generated interest in the recent renaissance in physics-based animation. O'Brien and colleagues [17] modeled ductile fracture by keeping track of plastic offsets that could be subtracted from the total deformation computed using a finite element method, essentially adding limited plasticity to a simulator designed for elastic solids. Irving and colleagues [10] introduced a multiplicative model of plasticity that is more appropriate for finite plastic deformations. Still, the amount of plastic deformation had to be limited to avoid inverting singular matrices. Bargteil and colleagues [1] allowed for large plastic flow by abandoning the initial rest configuration in favor of storing a rest state for every element. When rest states became ill-conditioned, the entire object was remeshed and simulation variables interpolated to the new mesh. A similar approach was employed by Wojtan and Turk [23] who incorporated a high-resolution surface mesh. Unfortunately, this wholesale remeshing leads to artificial diffusion due to resampling. Wicke and colleagues [22] addressed these problems by introducing the notion of an *embedded space*—a globally consistent embedding of the rest space into three dimensions. By performing local remeshing in the embedded space, artificial diffusion is limited to small errors exactly where plastic flow is occurring. Their approach allows for arbitrary elastic deformations and robustly handles extreme plastic deformations. Our own approach borrows their key idea, but to avoid complex remeshing strategies, we apply it in a point-based context where the equivalent operation is a simple neighborhood query in the embedded space.

An alternative approach to modeling elastoplasticity has focused on adding elastic behavior to fluid simulations [8, 12]. As one would expect, this approach easily handles arbitrary plastic flow, but demonstrates only limited elastic effects. However, this

approach is able to animate a variety of common materials, such as honey [18].

Researchers have also explored point-based animation of elastoplastic materials. In this case, moving least squares methods are used to estimate the deformation gradient and compute elastic forces. To handle limited plasticity, Müller and colleagues [16] introduced plastic offsets similar to those employed by O'Brien and colleagues [17]. For larger plastic deformations, they switched to a model closer to that of Goktekin and colleagues [8] that essentially stores elastic offsets that can be used to compute elastic forces. In their framework, materials can switch from one model to the other. Keiser and colleagues [11] introduced a unified framework by combining forces from smoothed particle hydrodynamics with elastic forces. Plasticity was modeled with additive plastic offsets, resulting in limitations on the amount of plastic flow. Solenthaler and colleagues [19] also adopted additive plastic offsets. Gerszewski and colleagues [7] developed a technique that tracked the deformation gradient through time, allowing the use of multiplicative plasticity and abandoning the storage of a rest state. This resulted in a unified framework for elastic solids and fluids, but, because they lack a rest state, they were only able to handle limited elastic deformations. Clavet and colleagues [4] modeled elastoplastic materials with a mass-spring system in which springs were dynamically inserted and removed. Their springs explicitly model elastic and viscous forces and include a model of plastic flow.

CHAPTER 3

RESEARCH

For more than two decades, computer graphics researchers have turned to the field of continuum mechanics for realistically animating deformable bodies. Our work makes use of several fundamental ideas from continuum mechanics. As a convenience to the reader, we present some background in the field prior to describing the specifics of our method. For a more thorough treatment, we refer the reader to the work of Fung [6].

3.1 Fundamental Concepts of Continuum Mechanics

At its core, the study of continuum mechanics is a study of how materials behave as they undergo displacement and deformation. As such, a sensible starting place for our discussion is the measuring of these quantities.

3.1.1 Deformation Gradient

A convenient means of expressing an object's deformation is to define a mapping from the object's reference state to its world state. The reference state is the shape the object would assume in the absence of all external forces, while the world state represents any configuration the shape might have assumed due to applied forces. We can express the mapping between the two states, or spaces, as:

$$\mathbf{x}(\mathbf{X}, t) = \mathbf{X}(t) + \mathbf{u}(\mathbf{X}, t), \tag{3.1}$$

where $\mathbf{x}$ represents the world space, $\mathbf{X}$ represents the reference space, and $\mathbf{u}$ is a vector field representing the displacement. All three are time dependent functions.

The measure of deformation around any arbitrary point in $\mathbf{X}$ is the Jacobian of this mapping with respect to changes in $\mathbf{X}$, which is given by

$$\nabla_{\mathbf{X}}\mathbf{x} = \mathbf{I} + \nabla_{\mathbf{X}}\mathbf{u} = \mathbf{F}. \tag{3.2}$$

This mapping, $\mathbf{F}$, captures both rigid body rotations and shape changing deformations (stretch and shear). If no displacement is present, $\mathbf{F}$ is clearly just the identity, $\mathbf{I}$.

3.1.2 Strain

Closely related to the deformation gradient, $\mathbf{F}$, is a quantity known as strain. Ideally, strain is a measure of how much the nonrotational portion of $\mathbf{F}$ deviates from the identity transformation. A simple and common formulation known as the Green-Saint-Venant strain is given as

$$\epsilon = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}). \tag{3.3}$$

If we were to write $\mathbf{F}$ in terms of its polar decomposition, $\mathbf{F} = \mathbf{RU}$, we can rewrite Equation (3.3) as

$$\begin{aligned} \epsilon &= \tfrac{1}{2}((\mathbf{RU})^T(\mathbf{RU}) - \mathbf{I}) \\ &= \tfrac{1}{2}(\mathbf{U}^T\mathbf{R}^T\mathbf{RU} - \mathbf{I}) \\ &= \tfrac{1}{2}(\mathbf{U}^T\mathbf{IU} - \mathbf{I}) \\ &= \tfrac{1}{2}(\mathbf{U}^T\mathbf{U} - \mathbf{I}). \end{aligned} \tag{3.4}$$

The rotational terms fall out of the equation by virtue of the fact that the transpose of a rotation matrix is its inverse. Thus, this strain metric is invariant to rotations and will only measure deformations that actually change the shape of the material.

Again rewriting Equation (3.3), this time in terms of Equation (3.2) and shortening $\nabla_{\mathbf{X}}\mathbf{u}$ to $\nabla\mathbf{u}$, yields

$$\begin{aligned} \epsilon &= \tfrac{1}{2}((\mathbf{I} + \nabla\mathbf{u})^T(\mathbf{I} + \nabla\mathbf{u}) - \mathbf{I}) \\ &= \tfrac{1}{2}((\mathbf{I}^2 + \nabla\mathbf{u} + \nabla\mathbf{u}^T + \nabla\mathbf{u}^T\nabla\mathbf{u}) - \mathbf{I}) \\ &= \tfrac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T + \nabla\mathbf{u}^T\nabla\mathbf{u}). \end{aligned} \tag{3.5}$$

Under very small amounts of deformation, the small values in $\nabla\mathbf{u}$ become even smaller in $\nabla\mathbf{u}^T\nabla\mathbf{u}$, and dropping this quadratic term out of the equation yields a fairly good approximation of the Green-Saint-Venant strain. This linearized strain is known as Cauchy's strain and can be written:

$$\begin{aligned} \epsilon &= \tfrac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \\ &= \tfrac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}. \end{aligned} \tag{3.6}$$

Unfortunately, Cauchy's strain is not invariant to rotations, which will lead to artifacts if it is used for large deformations.

3.1.3 Stress

Hooke's law for three-dimensional volumes states that stress, which is measured in units of force per unit area, depends linearly on the strain:

$$\sigma = \mathbf{C}\epsilon, \tag{3.7}$$

where $\mathbf{C}$ is a rank four tensor (the stiffness tensor) representing the constitutive model of the material. The stiffness tensor is comprised of 81 entries, though due to symmetries, it contains only 21 independent entries. Furthermore, if we can assume material isotropy, the number of independent entries reduces to two: Young's Modulus and Poisson's Ratio. These two values can be expressed in terms of the Lamé constants λ and μ, ultimately yielding the following equation for stress:

$$\sigma = \lambda \text{tr}(\epsilon)\mathbf{I} + 2\mu\epsilon. \tag{3.8}$$

3.1.4 Elastic Energy and Force

We define elastic energy density as:

$$\begin{aligned} \eta &= \tfrac{1}{2}(\epsilon \cdot \sigma) \\ &= \tfrac{1}{2}(\epsilon \cdot \mathbf{C}\epsilon). \end{aligned} \tag{3.9}$$

This energy is quadratic in ϵ, which is a measure of the material's deviation from the reference state. A quadratic function has no local minima, only a single global minimum. Taking a derivative of this energy term with respect to the displacement, $\mathbf{u}$, gives us a force that minimizes the strain, and thereby the stretch and shear displacements of the material, restoring the object to its reference state. This minimizing force can be written as

$$\begin{aligned} \mathbf{f} &= -\nabla_{\mathbf{u}}\eta \\ &= -\tfrac{1}{2}\nabla_{\mathbf{u}}(\epsilon \cdot \mathbf{C}\epsilon) \\ &= -\sigma\nabla_{\mathbf{u}}\epsilon. \end{aligned} \tag{3.10}$$

Once we have computed the force, we can integrate the force over time to compute the material's velocity and position at any time. In short, we can animate its motion.

3.2 Method Overview

The core of our approach is the maintenance of three domains: world space, embedded space, and rest space. *World space* refers to the current, deformed object configuration.

Embedded space is a global, least-squares best fit of the accumulated plastic deformation into three dimensions. It can be thought of as an approximate reference configuration for the material. *Rest space* is local to a particle and refers to the particle's preferred locations of its neighbors, or its preferred neighborhood orientation. Unlike world and embedded space, it cannot be encoded by storing a single position for each particle. Instead, we approximate the mapping from embedded to rest space with a per-particle linear transformation. We refer to these transformations as *plastic offsets*. Together these three spaces allow us to robustly define a mapping from rest space to world space, from which we can compute elastic forces and plastic deformations and account for changing local neighborhoods. Our method is summarized in Algorithm 1.

Algorithm 1 Deformation Embedding for Elastoplasticity

```
initialize kernel support radii and particle neighborhoods
while Animating do
  approximate deformation gradient
  factor deformation into elastic and plastic components
  compute elastic forces
  integrate elastic, body, and damping forces
  compute global embedding
  update neighborhoods
  compute local plastic offsets
end while
```

Figure 3.1 is an illustration of a possible state of the simulator. Globally, the particles disagree about their neighbors' positions. We resolve the inconsistencies through embedding and per-particle plastic offsets.

Notation: We use $\mathbf{x}_{ij}$, $\mathbf{e}_{ij}$, $\mathbf{u}_{ij}$ to denote *neighbor vectors*—vectors between particles i and j—in world, embedded, and rest space, respectively. Capital bold face letters refer to matrices.

3.3 Approximating Deformation

Following the work of Müller and colleagues [16], we use moving least squares (MLS) to approximate the deformation gradient, $\mathbf{F}$. For each particle i, we seek the matrix $\mathbf{F}_i$ that minimizes

$$\sum_j ||w_{ij}(\mathbf{F}_i\mathbf{u}_{ij} - \mathbf{x}_{ij})|| \tag{3.11}$$

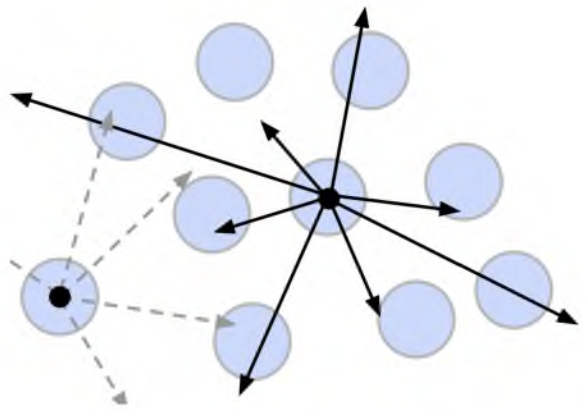

Figure 3.1. The embedded space of a material (circles) approximates the rest space (vectors) but cannot represent it exactly due to inconsistencies between particles.

where the sum is taken over the neighbors, j. The minimizer can be found by solving the following linear system,

$$\mathbf{F}_i(\sum_j w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T) = \sum_j w_{ij}\mathbf{x}_{ij}\mathbf{u}_{ij}^T, \tag{3.12}$$

which can be performed by inverting $\mathbf{A}_i = \sum_j w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T$. However, if the rest space near a particle is degenerate, i.e. neighboring particles are coplanar, colinear or nonexistent, $\mathbf{A}_i$ becomes singular and the simulation breaks down. Inspired by *oriented particles* [15], we assume that every particle occupies some finite volume in all directions. This assumption leads to adding $\lambda\mathbf{I}$ to $\mathbf{A}_i$, which in turn leads to adding $\lambda\mathbf{F}_i$ to both sides of Equation (3.12) yielding

$$\mathbf{F}_i(\lambda\mathbf{I} + \sum_j w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T) = \sum_j w_{ij}\mathbf{x}_{ij}\mathbf{u}_{ij}^T + \lambda\mathbf{F}_i \tag{3.13}$$

where λ is a positive constant, the magnitude of which is representative of the amount of volume intrinsic to a particle. This results in a basis matrix, $\mathbf{A}_i = \lambda\mathbf{I} + \sum_j w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T$, which is always invertible.

Because the $\mathbf{F}_i$ term on the right-hand-side is unknown, we approximate it using the deformation gradient from the previous timestep. The error introduced by this approximation is related to the change in $\mathbf{F}_i$ over a timestep, which we assume to be small. This technique works surprisingly well in practice—we have not noticed any artifacts, and our simulator is considerably more robust than if we were to use Equation (3.12).

The resulting computation of the deformation gradient is

$$\mathbf{F}_i = (\lambda\mathbf{F}_i^{prev} + \sum_j w_{ij}\mathbf{x}_{ij}\mathbf{u}_{ij}^T)\mathbf{A}_i^{-1}. \tag{3.14}$$

3.4 Particle Volume

It is common in point-based animation to assume that particles have a fixed, often uniform, mass or volume. This assumption often leads to density fluctuations that can cause spurious oscillations. In the context of plastic deformation, it is desirable to allow the sampling to vary over time and thus we require a way of estimating the volume occupied by a particle. Noting that $\mathbf{A}$ serves the same purpose as the basis matrix in tetrahedral finite element methods, that the volume of a tetrahedron is a multiple of the determinant of the basis matrix, and that the units of $\det(\mathbf{A})$ are m^6, we approximate the volume of a particle as

$$v_i = \sqrt{\frac{\det(\mathbf{A}_i)}{\lambda + (\sum_j w_{ij})^3}} \tag{3.15}$$

where λ is the value introduced in Equation (3.13). We initialize the value of λ to the volume of the sphere whose radius is half the global average neighbor distance at the start of the simulation. We subsequently hold the value of λ constant. As distances to neighbors get large, and as particles exit the neighborhood, the volume of particle i becomes λ.

We assign a constant density to the material and allow each particle's mass to change over time. The total mass of the material fluctuates slightly due to this approximation, but we did not observe any resulting artifacts in our examples.

This volume estimate permits uneven particle sampling as particles in coarser regions are assigned larger volumes than particles in dense regions. For example, it is possible to sample the object surface more densely than its interior as shown in Figure 3.2.

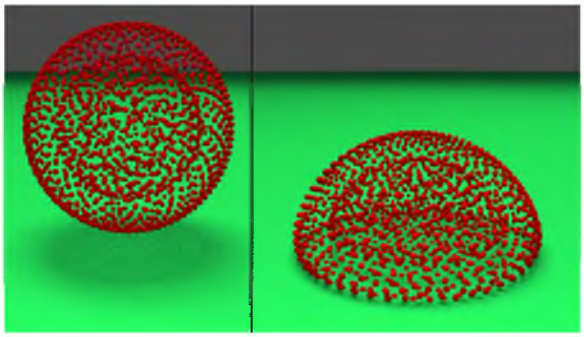

Figure 3.2. Deformation of an unevenly sampled sphere. Our volume estimate permits coarse sampling inside and dense sampling on the surface.

In Figure 3.3 we plot the percent change in volume of the unevenly sampled sphere as it undergoes plastic deformation. Over the course of 10 seconds of animation, the sphere loses a total of about 6 percent of its volume.

3.5 Elastoplasticity

Once we have found the deformation gradient, we must factor it into its elastic and plastic components. Upon computing the factorization, we use the elastic component to compute and integrate elastic forces. Subsequently we apply the plastic portion of the deformation to the rest shape of the material.

3.5.1 Deformation Factorization

We employ the plasticity model developed by Bargteil and colleagues [1] to factor the deformation gradient, $\mathbf{F}$, into elastic and plastic components:

$$\mathbf{F} = \mathbf{F}_e \cdot \mathbf{F}_p. \tag{3.16}$$

Prior plasticity models in computer graphics had used an additive strain decomposition, which is only physically meaningful under infinitesimal deformation. A multiplicative deformation decomposition, as noted by Irving and colleaques [10], allows a complete

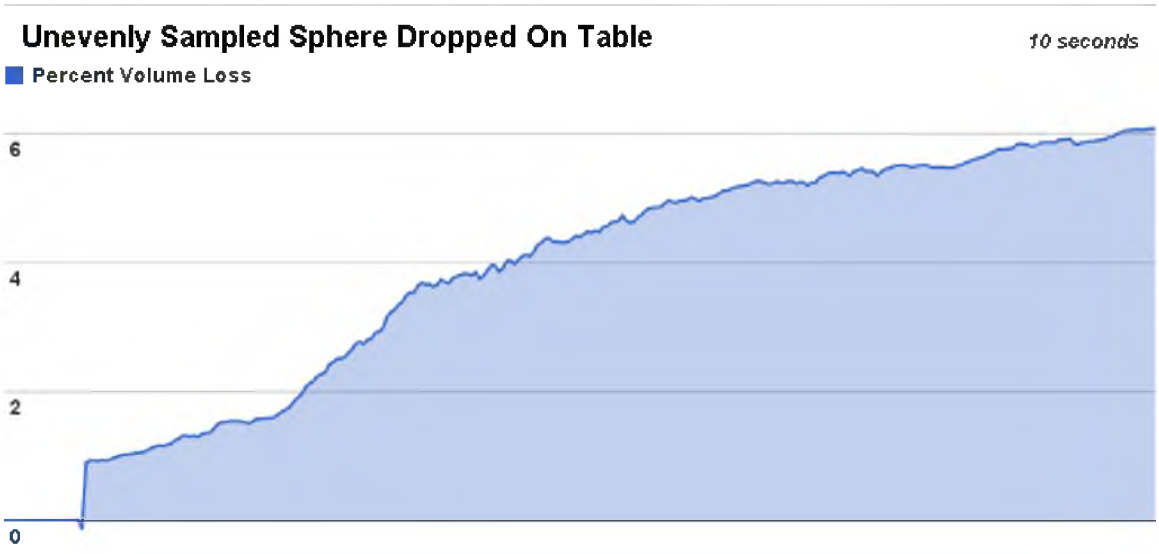

Figure 3.3. Percent change in volume of the unevenly sampled sphere featured in Figure 3.2 undergoing plastic deformation.

separation of plastic flow and elastic forces. Additionally, by choosing $\mathbf{F}_p$ to have a determinant of 1, we can easily enforce volume preserving plastic flow.

To compute the plastic deformation in the neighborhood of a single particle, we first diagonalize the deformation gradient using the singular value decomposition:

$$\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T. \tag{3.17}$$

The matrices $\mathbf{U}$ and $\mathbf{V}$ are rotations and $\hat{\mathbf{F}}$ is diagonal. Making use of $\hat{\mathbf{F}}$, we compute

$$\hat{\mathbf{F}}^* = (\det(\hat{\mathbf{F}}))^{-1/3}\hat{\mathbf{F}}, \tag{3.18}$$

where $\det(\hat{\mathbf{F}}^*) = 1$ and thus preserves volume. We then compute

$$\hat{\mathbf{F}}_p = (\hat{\mathbf{F}}^*)^\gamma, \tag{3.19}$$

where γ is the following function of the current stress ($\mathbf{S}$), the yield stress limit (S_Y), the flow rate (ν), and hardening parameters (α, K):

$$\gamma(\mathbf{S}, S_Y, \nu, \alpha, K) = \Delta t \nu \frac{\|\mathbf{S}\| - S_Y - K\alpha}{\|\mathbf{S}\|}. \tag{3.20}$$

This function, clamped to values between 0 and 1, represents the amount of deformation that is absorbed into the rest shape in a timestep Δt. The parameter α serves as an accumulator for plastic stress over time and is updated at every timestep as follows:

$$\dot{\alpha} = \|\mathbf{S}\|. \tag{3.21}$$

The hardening parameter K, depending on its sign, effectively serves either to raise or lower the yield stress limit S_Y, enabling either work hardening or work softening, respectively.

3.5.2 Elasticity

Factoring $\mathbf{F}_e$ as in Equation (3.17), and plugging in to Equation (3.6), we can compute a diagonalized Cauchy strain that is invariant to rotations. The following equation simplifies due to the symmetric nature of $\hat{\mathbf{F}}_e$:

$$\begin{aligned} \hat{\epsilon} &= \tfrac{1}{2}(\hat{\mathbf{F}}_e + \hat{\mathbf{F}}_e^T) - \mathbf{I} \\ &= \hat{\mathbf{F}}_e - \mathbf{I}. \end{aligned} \tag{3.22}$$

As noted by Irving and colleagues [10], the diagonalization of $\mathbf{F}_e$, results in the diagonalization of stress. Thus, making use of the diagonalized Cauchy strain, as well as Equation (3.8), we compute diagonalized stress:

$$\hat{\sigma} = \lambda \text{tr}(\hat{\mathbf{F}}_e - \mathbf{I}) + 2\mu(\hat{\mathbf{F}}_e - \mathbf{I}). \tag{3.23}$$

We then rotate the diagonalized stress $\sigma = \mathbf{U}\hat{\sigma}\mathbf{V}^T$ and following Müller and colleagues [16], compute symmetrized forces:

$$\begin{aligned} \mathbf{f}_i &= 2v_i\sigma_i\mathbf{A}_i^{-1}w_{ij}\mathbf{u}_{ij} \\ \mathbf{f}_j &= -\mathbf{f}_i, \end{aligned} \tag{3.24}$$

which are linear in position.

For damping forces, we use the SPH viscosity formulation of Müller and colleagues [14]. To update positions and velocities, we use the modified Newmark-beta integration scheme proposed by Bridson and colleagues [3], which integrates viscous forces implicitly and elastic forces explicitly:

- $v^{n+1/2} = v^n + \frac{\Delta t}{2} a_{viscous}(t^n, x^n, v^{n+1/2})$
- $x^{n+1} = x^n + \Delta t v^{n+1/2}$
- $v^{n+1} = v^{n+1/2} + \frac{\Delta t}{2} a_{elastic,body}(t^{n+1}, x^{n+1}, v^{n+1/2})$

This is a second order method. The first half-velocity step is integrated implicitly, while the position update and the second half-velocity step are integrated explicitly. As noted by Bridson and colleagues, by performing the implicit portion of the integration only on the viscous forces, any artificial damping introduced by the integration is added to the portion of the system where there is damping anyway. No artificial damping is introduced into the elastic modes of the system.

3.6 Embedding Deformation

To handle plastic flow, we maintain a globally optimal embedded space, encoded as a three-dimensional position, $\mathbf{e}_i$, for each particle. We also update local plastic offsets to rest space, encoded as a 3×3 transformation matrix, $\mathbf{P}_i$, for each particle. We now describe how these are computed. First, we compute temporary rest space vectors that incorporate the plastic deformation that occurred over the previous timestep, $\tilde{\mathbf{u}}_{ij} = \mathbf{F}_p\mathbf{u}_{ij}$. The optimal embedding of the particles into three-dimensional space will minimize the

differences of neighbor vectors between the embedded and rest spaces. We formulate this optimization as a weighted linear least-squares problem

$$\underset{\mathbf{e}_i}{\operatorname{argmin}} \sum_{i,j} ||w_{ij}(\tilde{\mathbf{u}}_{ij} - \mathbf{e}_{ij})||. \tag{3.25}$$

We solve this over-constrained problem using the normal equations,

$$\mathbf{K}^T\mathbf{K}\mathbf{e} = \mathbf{K}^T\mathbf{u}. \tag{3.26}$$

The vector $\mathbf{e}$ contains the embedded space positions for all particles. For each particle i and neighbor j, $\mathbf{K}$ contains three rows corresponding to the operation $w_{ij}(\mathbf{e}_j - \mathbf{e}_i)$ and the corresponding entry in $\mathbf{u}$ contains $w_{ij}\tilde{\mathbf{u}}_{ij}$. $\mathbf{K}$ contains three rows constraining the first particle to maintain its current embedded position, preventing arbitrary translations. We solve this linear system with conjugate gradients and do not explicitly compute $\mathbf{K}^T\mathbf{K}$.

While these embedded space positions provide a globally consistent reference configuration, there will still be differences between the rest and embedded spaces. To account for this, we compute a local fit per particle similar to the "plastic offsets" of Wicke and colleagues [22]. Here we seek the best mapping, $\mathbf{P}_i$, from embedded space to rest space for each particle, which we again compute using a weighted least squares solve. Here we minimize

$$\underset{\mathbf{P}_i}{\operatorname{argmin}} \sum_{j} ||w_{ij}(\mathbf{P}_i\mathbf{e}_{ij} - \tilde{\mathbf{u}}_{ij})|| \tag{3.27}$$

We use a regularization similar to that in Equation (3.14) to handle degenerate embedded space positions and obtain

$$\mathbf{P}_i = (\mathbf{P}_i^{prev} + \sum_j w_{ij}\tilde{\mathbf{u}}_{ij}\mathbf{e}_{ij}^T)(\mathbf{I} + \sum_j w_{ij}\mathbf{e}_{ij}\mathbf{e}_{ij}^T)^{-1} \tag{3.28}$$

We do not store rest space vectors, and instead compute them as needed by $\mathbf{P}_i\mathbf{e}_{ij}$.

3.7 Updating Neighborhoods

After plastic flow, a particle's neighbors may have moved far away, and no longer provide useful information about the deformation gradient. We therefore update each particle's neighborhood by finding the nearest neighbors in the embedded space. We expect that neighborhoods in embedded space are a good approximation of neighborhoods in rest space, and they can be queried efficiently using a KD-tree. In order to avoid popping artifacts, we fade neighbors in and out over several timesteps by scaling kernel weights when neighbors enter or exit a particle's neighborhood.

3.8 Implementation Details

For our weighting kernel, we use the standard SPH Poly6 kernel of Müller and colleagues [14],

$$W_{\text{poly6}}(r,h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - r^2\right)^3 & 0 \leq r \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{3.29}$$

where r is the distance between a pair of particles, and h is the support radius. This kernel has local support, and because it is a function of the square of the distance between particles, it is not necessary to perform a square root in the distance computations.

We initialize each particle's radius by finding its nearest 32 neighbors, and setting its radius to twice the average neighbor distance. During the simulation, a particle can have a maximum of 32 neighbors, which must be within its radius in embedded space.

CHAPTER 4

RESULTS AND FUTURE WORK

We have used our proposed method to simulate materials exhibiting a wide range of elastoplastic behavior as shown in our figures. Timing results are shown in Table 4. All examples were run on a single core of a 2.8 GHz Intel Xeon processor. Rendering surfaces were generated using the particle skinning method of Bhattacharya and colleagues [2]. All timestep sizes listed were experimentally determined to be the largest one can take for stability.

Figure 4.1 shows a block falling down a set of stairs into a pool. The material undergoes work softening; initially it bounces off of the obstacles, but eventually it oozes out into the pool. This example demonstrates the ability of our method to handle both elastic bounces and extreme plastic flow.

Figure 4.2 compares the world space and embedded space deformations of a set of cubes dropped on a bar. The embedded space captures the key features of the global plastic deformation. Again, we note the wide range of material parameters that can be simulated using our approach. Similar results for cubes dropped on flat ground are shown in Figure 4.3.

Figure 4.4 shows the simulated particles of a high-resolution bunny along the bottom row, with corresponding surfaces along the top row.

Like most elastoplastic simulators, remaining stable during large plastic flows is a significant challenge for our method. Because we guarantee $\mathbf{F}_p$ to be volume preserving, plastic deformation in regions that are becoming planar can result in particles being pushed away in an unstable manner. Typically when simulations become unstable, it is due to particles gaining extreme velocities in regions that become very thin in embedded space. One possible solution to these instabilities is to introduce a regularization term in our embedding solve to prevent sudden changes in the embedding. In practice, reducing the timestep increases stability at the cost of longer running times.

Table 4.1. Timing results for pictured examples

Example	#Particles	Δt	sec/frame
Figure 3.2	1415	.0001	17
Figure 4.1	5488	.0002	105
Figure 4.2 (left)	8000	.0005	115
Figure 4.2 (center)	8000	.0005	95
Figure 4.2 (right)	8000	.0005	51
Figure 4.4	16525	.00005	536

Although our novel volume estimation technique permits us to use uneven sampling, we have not thoroughly explored the space of sampling strategies. More work is necessary to understand how changes in sampling rates affect stability. In addition, dynamically resampling the volume is an obvious direction for future work.

It is common for materials to become brittle after work hardening and become prone to fracture. In our implementation, such topological changes happen by accident, when particle neighborhoods change and parts of the material stop interacting. By intentionally causing such topological changes, our method could be adapted to simulate phenomena such as ductile fracture.

Another opportunity for future work would be to extend our approach to the *Elastons* framework of Martin and colleagues [13]. This direction seems particularly promising as the use of derivative information could address difficulties with degenerate particle neighborhoods.

We have presented a simple to implement point-based method for animating elasto-plastic materials by maintaining a globally optimal fit of the material's reference configuration. This approach allows us to simulate materials undergoing arbitrary elastic deformations as well as extreme plastic flows without expensive and complex remeshing operations. In addition, we have introduced robustness improvements and a new volume estimation technique that can improve existing point-based techniques.

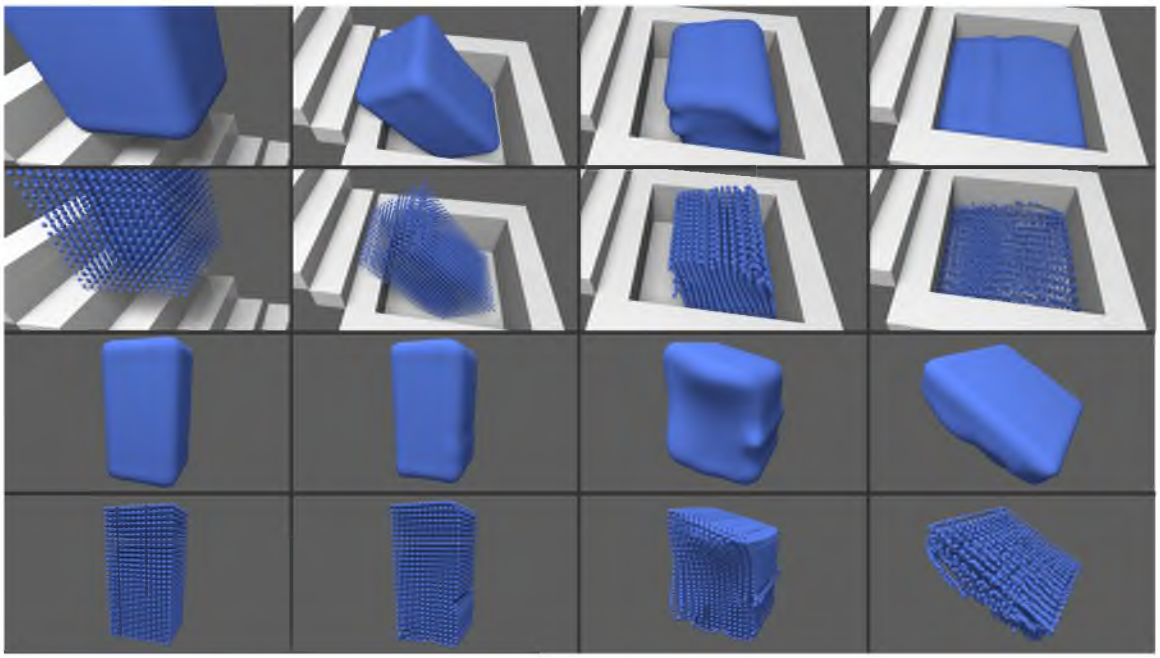

Figure 4.1. A tower undergoes plastic deformation with work softening as it falls down stairs into a pool. The embedded space is shown below.

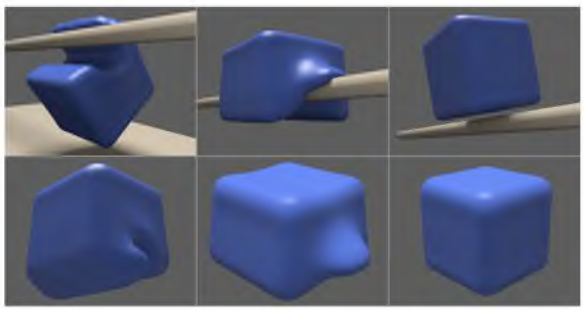

Figure 4.2. World space (above) and embedded space (below) of cubes with varying material parameters dropped on a rigid bar

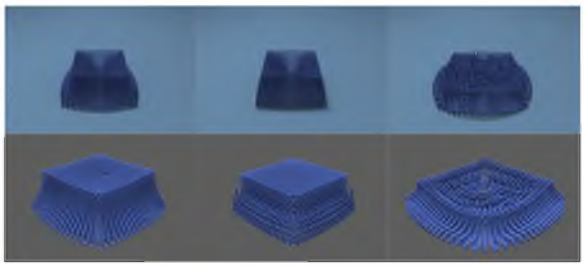

Figure 4.3. World space (above) and embedded space (below) of cubes with varying material parameters dropped on flat ground

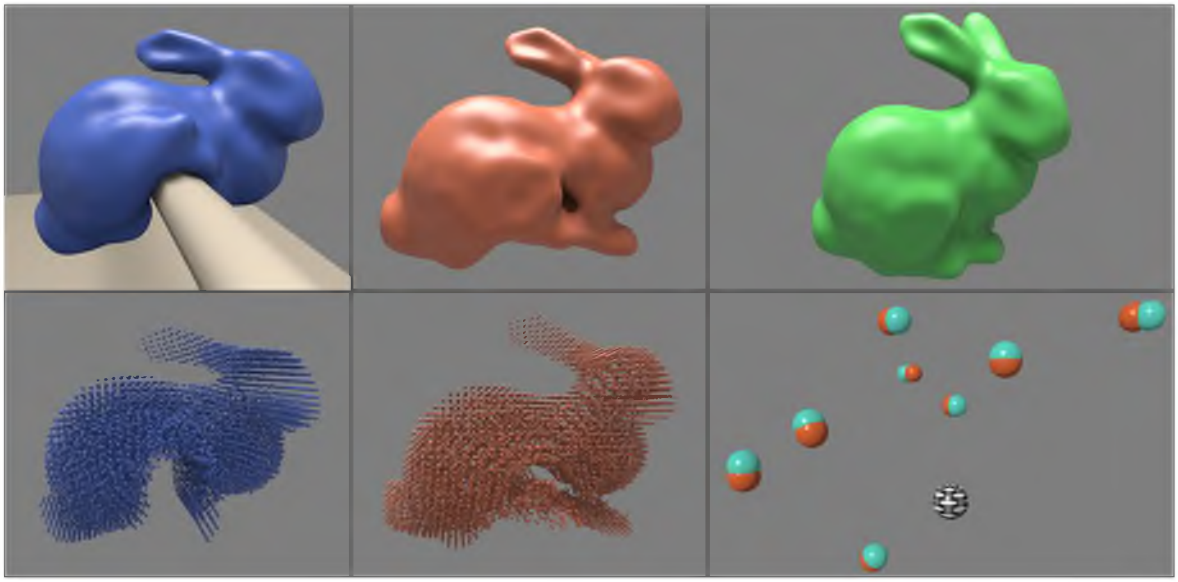

Figure 4.4. A bunny falls on a rod and undergoes elastic and plastic deformation. The embedded space (center column) captures much of the deformation from the initial configuration (top right), but does not quite match the teal rest space positions (bottom right). World space (left column) shows additional elastic deformation compared to embedded space.

REFERENCES

[1] Bargteil, A. W., Wojtan, C., Hodgins, J. K., and Turk, G. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. 26*, 3 (2007), 16.

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

[3] Bridson, R., Marino, S., and Fedkiw, R. Simulation of clothing with folds and wrinkles. In *The Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 28–36.

[4] Clavet, S., Beaudoin, P., and Poulin, P. Particle-based viscoelastic fluid simulation. In *The Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 219–228.

[5] Fleischer, K., Witkin, A., Kass, M., and Terzopoulos, D. Cooking with kurt. *Animation in ACM SIGGRAPH Video Review*, 36 (1987).

[6] Fung, Y. C. *A First Course in Continuum Mechanics.* Prentice-Hall, Englewood Cliffs, N.J., 1969.

[7] Gerszewski, D., Bhattacharya, H., and Bargteil, A. W. A point-based method for animating elastoplastic solids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aug 2009).

[8] Goktekin, T. G., Bargteil, A. W., and O'Brien, J. F. A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 3 (2004), 463–468.

[9] Goktekin, T. G., Reisch, J., Peachey, D., and Shah, A. An effects recipe for rolling a dough, cracking an egg and pouring a sauce. In *ACM SIGGRAPH 2007 sketches* (2007), p. 67.

[10] Irving, G., Teran, J., and Fedkiw, R. Invertible finite elements for robust simulation of large deformation. In *The Proceedings of the ACM/Eurographics Symposium on Computer Animation* (2004), pp. 131–140.

[11] Keiser, R., Adams, B., Gasser, D., Bazzi, P., Dutré, P., and Gross, M. A unified Lagrangian approach to solid-fluid animation. In *The Proceedings of the Symposium on Point-Based Graphics* (2005), pp. 125–133.

[12] Losasso, F., Shinar, T., Selle, A., and Fedkiw, R. Multiple interacting liquids. *ACM Trans. Graph. 25*, 3 (2006), 812–819.

[13] Martin, S., Kaufmann, P., Botsch, M., Grinspun, E., and Gross, M. Unified simulation of elastic rods, shells, and solids. *ACM Trans. Graph. 29* (2010), 39:1–39:10.

[14] Müller, M., Charypar, D., and Gross, M. Particle-based fluid simulation for interactive applications. In *The Proceedings of the Symposium on Computer Animation* (2003), pp. 154–159.

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

[16] Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., and Alexa, M. Point based animation of elastic, plastic and melting objects. In *The Proceedings of the ACM/Eurographics Symposium on Computer Animation* (2004), pp. 141–151.

[17] O'Brien, J. F., Bargteil, A. W., and Hodgins, J. K. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21*, 3 (2002), 291–294.

[18] Ruilova, A. Creating realistic cg honey. In *ACM SIGGRAPH 2007 posters* (New York, NY, USA, 2007), ACM, p. 58.

[19] Solenthaler, B., Schläfli, J., and Pajarola, R. A unified particle model for fluid-solid interactions. *Journal of Visualization and Computer Animation 18*, 1 (2007), 69–82.

[20] Terzopoulos, D., and Fleischer, K. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *The Proceedings of ACM SIGGRAPH* (1988), pp. 269–278.

[21] Terzopoulos, D., Platt, J., and Fleischer, K. Heating and melting deformable models (from goop to glop). In *The Proceedings of Graphics Interface* (1989), pp. 219–226.

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

[23] Wojtan, C., and Turk, G. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27* (2008), 47:1–47:8.