Copyright © 2010 Tech Science Press CMES, vol.63, no.2, pp.117-136, 2010

Enhanced Understanding of Particle Simulations Through Deformation-Based Visualization

A.N.M. Imroz Choudhury[1], Michael D. Steffen[1], James E. Guilkey[2]
Steven G. Parker[3]

Abstract: We present a physically based method for visualizing deformation in particle simulations, such as those describing structural mechanics simulations. The method uses the *deformation gradient* tensor to transform carefully chosen glyphs representing each particle. The visualization approximates how simulated objects responding to applied forces might look in reality, allowing for a better understanding of material deformation, an important indicator of, for example, material failure. It can also help highlight possible errors and numerical deficiencies in the simulation itself, suggesting how simulations might be changed to yield more accurate results.

Keywords: Deformation, visualization, particle methods, material point method.

1 Introduction and Background

Particle methods [Belytschko, Krongauz, Organ, Fleming, and Krysl (1996)] are numerical simulation methods in which materials are modeled by collections of discrete computational particles, which can move about the computational domain as indicated by the model equations. Generally speaking, these methods produce output in which each particle is identified by its location and additional data values. The values can be of any type, and methods producing scalar, vector, and tensor values are common. Compared to the well-known finite element method (FEM) [Bathe (1996)], particle methods have the major advantage that they are well-suited to handling large deformations, such as might be found in simulations leading to material failure, for example. When dealing with such deformations, FEM can suffer from entangled, inverted, or otherwise ill-conditioned meshes; remeshing can alleviate these problems but only at heavy cost, both in terms of performance

[1] SCI Institute, Salt Lake City, UT, USA.
[2] Schlumberger Technology Corporation, Salt Lake City, UT, USA (corresponding author).
[3] NVIDIA Corporation, Salt Lake City, UT, USA.

and accuracy. For simulations in which large deformations are expected, as in biomechanics, or where typical engineering materials will experience failure, such as explosions, particle methods, which are not affected by these concerns, can be a suitable alternative.

Because the particles “carry” data values, a glyph-based approach to visualiztion that places geometry at the position of each particle is suitable. Scalar data values can be displayed by varying free parameters, such as glyph size and color. However, it is not clear how to visualize deformation within this framework, an important quantity in structural mechanics that conveys information about the strains and resulting stresses experienced by an object of study.

Deformation is an important physical response to loading, and an independent variable in models of material failure [Maloney and Lemaître (2004); Meakin (1991)]; as such it is an important consideration in simulations involving large deformations, such as for penetrative tissue damage [Ionescu, Guilkey, Berzins, Kirby, and Weiss (2006)], high-temperature damage in pipes [Hall and Hayhurst (1991)], and explosion in containers of high-energy materials [Guilkey, Harman, and Banerjee (2007)]. Because deformation is directly observable by humans in physical objects, it should be included in visualization of such simulations, at the very least as an indirect way to validate simulation data. Leaving deformation out of the process produces at best a deficient visualization, and at worst, a misleading one.

In their common usage, glyph-based visualization approaches force users to infer deformation from the relative motion of the glyphs, and for many arrangements, the actual deformation can be difficult to discern. This paper presents a particular approach to glyph-based visualization for particle data that uses carefully chosen glyphs and the *deformation gradient* to clearly display deformation, both at local and global scales. We demonstrate how the method improves on current techniques, allowing scientists a better understanding of their simulations.

1.1 Particle Methods

Particle methods form a subset of the mesh free [Liu (2003)] or meshless [Belytschko, Krongauz, Organ, Fleming, and Krysl (1996); Atluri, Liu, and Han (1998)] methods, which differ from FEM (and other fully meshed methods such as finite differences, etc.) in that no object geometry is represented by a mesh. Particle methods represent objects by discretizing them into collections of particles, each of which is a Lagrangian representation of some part of the object either directly as a small continuum of matter, or a sampling point. Particles are, in principle, free to move independently about the domain, but they are usually restricted to realistic behavior by material models and other physical constraints.

Examples of particle methods include smoothed particle hydrodynamics (SPH) [Monaghan (2005)], the related smooth particle applied mechanics [Hoover (2006)], and the particle-in-cell (PIC) family of methods [Harlow (1964); Brackbill and Ruppel (1986)], in which a stationary, background mesh is used to compute gradients and the particles move through the grid cells as the simulation proceeds. The material point method (MPM) [Sulsky, Chen, and Shreyer (1994); Sulsky, Zhou, and Schreyer (1995)] (along with its successor, the generalized interpolation material point method (GIMP) [Bardenhagen and Kober (2004)]) is a PIC method explicitly designed for performing structural mechanics simulations. MPM represents geometry by discretizing objects into particles, or *material points*. Each material point in such a particle model represents a small piece of material from the object and obeys the laws of continuum mechanics. Each material point carries several physical parameters, such as mass, volume, stress, etc. In the simulation, the particles move as the object responds to loads placed on it, resulting in an overall deformation of the object. In the process of updating particle volumes, the MPM algorithm may compute a tensor quantity known as the *deformation gradient*, which acts as a local measure of distortion affecting each particle. This value may be used as input to the material constitutive model, which relates stress to strain; however our method takes advantage of the deformation gradient to visualize deformation.

1.2 Overview of MPM/GIMP

Numerous flavors exist of both standard MPM and GIMP. Particular choices of grid and particle basis functions [Steffen, Wallstedt, Guilkey, Kirby, and Berzins (2008)], whether or not to lump the mass matrix [Love and Sulsky (2006)], and what time-stepping algorithm to use (implicit, explicit, etc.) [Guilkey and Weiss (2003); Wallstedt and Guilkey (2008)] will have large impacts on implementation details as well as expected algorithm performance. Here, we present a very brief overview of a single timestep in the Update Stress Last (USL) explicit method. We do this to illustrate the role that the deformation gradient plays in the method. For full implementation details, we suggest referring to Wallstedt and Guilkey (2008).

The simulation begins with a collection of particles approximating the geometry of the objects of interest (Figure 1). Each particle is assigned various initial quantities: mass (m_p), position ($\mathbf{x}_p^0$), velocity ($\mathbf{v}_p^0$), deformation gradient ($\mathbf{F}_p^0$), volume (V_p^0), as well as other quantities pertaining to the particular constitutive model used (temperature, plasticity states, etc.). To advance the simulation from time t^k to time $t^{k+1} = t^k + \Delta t$, a Galerkin projection of particle momenta to a (usually Cartesian) grid is first carried out, allowing grid velocity to be calculated. This projection is

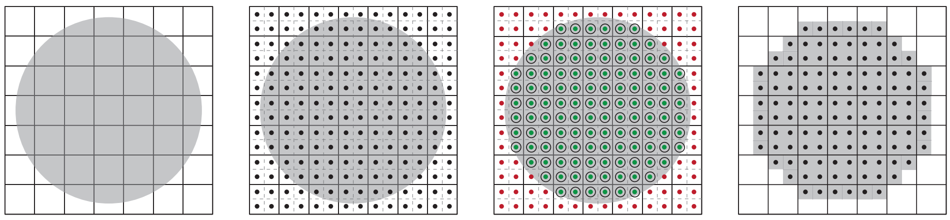

Figure 1: A particle model of a disc is created. (a) The gray area represents the disc, laid on top of the MPM background grid. (b) The grid cells are subdivided into subcells, and the subcell centroids are sampled. (c) The circled samples fall inside the disc and become particles in the model. (d) The particles are shaped like the subcells they were sampled from, so the final model has unavoidably jagged edges.

approximated in MPM as:

$$m_i = \sum_p \phi_i(\mathbf{x}_p) m_p, \tag{1}$$

$$\mathbf{v}_i^k = \frac{\sum_p \phi_i(\mathbf{x}_p^k)\mathbf{v}_p^k m_p}{m_i}, \tag{2}$$

where m_i, $\mathbf{v}_i$, ϕ_i are the grid mass, grid velocity, and grid basis functions, respectively. Next, the internal force is computed at the nodes as an approximated volume integral of the divergence of particle stress:

$$\mathbf{f}_i^{int} = -\sum_p \nabla\phi_i(\mathbf{x}_p^k) \cdot \boldsymbol{\sigma}_p^k V_p^k. \tag{3}$$

External forces (body forces and tractions) are specified on the grid (or projected from particles to the grid), and grid acceleration is computed as:

$$\mathbf{a}_i = \frac{\mathbf{f}_i^{int} + \mathbf{f}_i^{ext}}{m_i}. \tag{4}$$

An updated grid velocity is found with a Forward Euler scheme:

$$\mathbf{v}_i^* = \mathbf{v}_i^n + \mathbf{a}_i \Delta t. \tag{5}$$

Next, particle positions and velocities are updated by evaluating the resulting grid velocity and acceleration fields at the particle locations:

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \sum_i \phi_i(\mathbf{x}_p^k)\mathbf{v}_i^* \Delta t, \tag{6}$$

$$\mathbf{v}_p^{k+1} = \mathbf{x}_p^k + \sum_i \phi_i(\mathbf{x}_p^k)\mathbf{a}_i\Delta t. \tag{7}$$

Evaluation of the gradient of the grid velocity function provides a velocity gradient at particle positions.

$$\nabla\mathbf{v}_p = \sum_i \nabla\phi_i(\mathbf{x}_p^{k+1})\mathbf{v}_i^*. \tag{8}$$

This velocity gradient is used in updating particle deformation gradient, which is in turn used in calculating particle stress and particle volume:

$$\mathbf{F}_p^{k+1} = (\mathbf{I} + \nabla\mathbf{v}_p\Delta t)\mathbf{F}_p^k, \tag{9}$$

$$\boldsymbol{\sigma}_p^{k+1} = \boldsymbol{\sigma}(\mathbf{F}_p^{k+1}), \tag{10}$$

$$V_p^{k+1} = V_p^0 \det(\mathbf{F}_p^{k+1}), \tag{11}$$

where the constitutive model relating deformation to stress is represented by the general function $\boldsymbol{\sigma}$. Note that while Equation 10 has a dependence only on deformation gradient, implying a hyperelastic material model, the methodology described here is completely general with respect to constitutive model. In other words, while it *is* necessary to compute a value for $\mathbf{F}_p$ as given in Equation 9, any constitutive model (e.g., a hypoelastic model with rate dependent plasticity) may be used. Even if the deformation gradient is not used to advance the solution, advancing it in time is of low cost, and, as the examples below will show, high value. In addition, of course, the deformation gradient is valuable in that it can be used to compute numerous measures of strain.

Equations (1 - 11) outline a single timestep of MPM. GIMP follows the same outline, but the terms $\phi_i(\mathbf{x}_p)$ are replaced by a different function $\overline{\phi}_{ip}$, calculated as a convolution of the grid basis function ϕ_i and a particle characteristic function χ_p (for more details about GIMP and particle characteristic functions, see Bardenhagen and Kober (2004)).

In the remainder of this paper, we will be working with GIMP simulations, though the method we present works with traditional MPM. It will also work with any particle method that treats particles as subvolumes of a continua (as opposed to sampling points) that is capable of computing a deformation gradient, whether or not it is used to advance the solution.

2 Related Work

Bigler, Guilkey, Gribble, Hansen, and Parker (2006) present an overview of their methods for visualizing MPM data produced by the Center for the Simulation of

Accidental Fires and Explosions project (C-SAFE) [Henderson, McMurtry, Smith, Voth, Wight, and Pershing (2000)]. They use the SCIRun Problem Solving Environment [Parker and Johnson (1995)] to view smaller data sets using standard graphics hardware, and a high-performance ray tracer [Bigler, Stephens, and Parker (2006)] to handle the millions of primitives in larger data sets. In both systems, each particle is represented by a sphere of a specific color and radius, both chosen to represent physical parameters in the data set. Generally, in such visualizations deformation is understood on a global scale, allowing the particle positions to relay an idea of deformation across the whole model, without any indication of the shapes of individual particles. For understanding smaller structures at closer scales, lighting and non-photorealistic rendering techniques are employed. Shadowing and ambient occlusion give visual cues about relative positions of ambiguously oriented particles, while silhouette edges help to bring attention to medium-scale structures by highlighting closely bound groups of particles.

Gribble, Stephens, Guilkey, and Parker (2006) developed a visualization system that takes advantage of the symmetry of spheres. They use programmable graphics hardware to accelerate particle rendering by using a texture mapped billboard of a single sphere to represent each particle, instead of rendering actual geometry. By reducing the number of triangles needed to just two per billboard, the system achieves interactive rates on desktop machines for order-of-million-particle data sets. The standard method of simply rasterizing millions of triangle-tessellated spheres quickly overwhelms current graphics hardware; this approach is, therefore, notable for its high performance. However, as it displays the same prerendered sphere geometry for every particle, it cannot handle deformation data via glyph-based tensor visualization as discussed above.

In summary, state-of-the-art particle data visualization treats each particle simply as a position associated with data. This simple approach works well for many purposes, but this paper will argue that we can build on these methods, including deformation data in a physically correct and visually insightful way.

On the other hand, the graphics community's treatment of deformation tends toward approximate but credible images rather than numerical accuracy, since such approximations are computationally cheaper to achieve. Such methods are used in settings where accuracy is secondary to visual effect, such as in movies and video games. Typical methods are based on simulation of continuum and fracture mechanics through finite elements for both brittle [O'Brien and Hodgins (1999)] and ductile [O'Brien, Bargteil, and Hodgins (2002)] fracture. Irving, Teran, and Fedkiw (2004) have developed invertible finite elements as an extension to standard finite elements which behaves robustly enough to handle physical situations with extreme mesh deformations and even inverted mesh elements, but at a significant cost in the accuracy of the results. These methods are valuable mainly for visual results and not scientific accuracy. These and other related methods in graphics can be traced back to methods dealing both with elasticity [Terzopoulos, Platt, Barr, and Fleischer (1987)] and inelasticity (i.e. viscoelasticity, plasticity, and fracture) [Terzopoulos and Fleischer (1988)] in graphical models.

3 Visualizing Particle Deformation

Glyph-based particle visualization methods can be summarized as follows: For each particle p situated at location $\mathbf{x}_p$, select a set of points G_p (representing some glyph geometry) centered at the origin, a deformation operator $\mathbf{D}_p$, and a color C_p. Render the deformed geometry $\mathbf{D}_p(G_p)$ translated to location $\mathbf{x}_p$ with color C_p.

Common particle visualization techniques typically use a unit sphere for G_p, a scaling operation for $\mathbf{D}_p$ with a magnitude chosen to reflect a scalar data value on particle p (or alternatively, simply no scaling at all), and a scalar color map applied to another value from p to specify C_p. These choices yield spheres whose radii possibly reflect one scalar value (commonly a volume or mass), and whose colors reflect another (such as equivalent stress, velocity magnitude, temperature, etc.).

In order to visualize deformation, we instead use $\mathbf{F}_p$, the deformation gradient of particle p, for the deformation operator by applying it as a linear transformation to the points making up the origin-centered glyph. This transformation illustrates the local deformation by applying it directly to the piece of the object on which it acts.

The key is to select a glyph geometry that communicates information about the deformation gradient. For example, spheres do not work for the simple reason that they cannot indicate pure rotation. Suppose G_S is a unit sphere centered at the origin, and $\mathbf{F}$ is a deformation gradient with polar decomposition [Strang (1988)] $\mathbf{F} = \mathbf{RS}$ (in which $\mathbf{R}$ is orthonormal, representing a rotation, and $\mathbf{S}$ is symmetric and positive-definite, representing non-uniform but orthogonal scaling). Because $\mathbf{S}$ is symmetric and positive-definite, it can be decomposed as $\mathbf{S} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$, where $\boldsymbol{\Lambda}$ is diagonal and $\mathbf{Q}$ is orthonormal. Because rotational transforms do not affect a sphere, we also have $\mathbf{U}G_S = G_S$ for any orthonormal $\mathbf{U}$.[1] Finally, concatenating rotation transforms (represented by products of orthonormal matrices) yields a rotation transform; i.e. $\mathbf{RQ}$ can be written as $\mathbf{U}$ for orthonormal $\mathbf{R}$ and $\mathbf{Q}$, and some

[1] We have adopted the notation $\mathbf{F}S$ for the notion of a tensor $\mathbf{F}$ acting on a set of points S, which we define as follows: $\mathbf{F}S \equiv \{\mathbf{F}x | x \in S\}$.

orthonomal **U**:

$$\begin{aligned}
\mathbf{F}G_S &= \mathbf{RS}G_S \\
&= \mathbf{RQ\Lambda Q}^T G_S \\
&= \mathbf{RQ\Lambda Q}^T \mathbf{R}^T G_S \\
&= \mathbf{U\Lambda U}^T G_S \\
&= \mathbf{S}' G_S.
\end{aligned}$$

The general transform **F** is aliased by some symmetric positive-definite transform $\mathbf{S}'$. In other words, with respect to spheres, *every linear transformation is a scaling transformation.* If the tensors used for deforming the glyphs are symmetric and positive-definite (such as stress), then they lack a rotational component and spheres (or other glyphs [Kindlmann (2004)]) can be used illustrate the tensors' eigenspaces. For purposes of studying deformation, however, capturing rotation is necessary, so a more suitable glyph geometry is required.

In addition to properly handling rotation, the glyph geometry should also represent the initial discretization behind the structural mechanics simulation. Often spheres are used to visualize particle data because particles often represent geometric points. The word "point" in "material point method" emphasizes this simplifying idea. However, MPM models continuum mechanics, and so rather than being dimensionless points, MPM particles represent computational volumes that initially partition a continuous object.

The common choice of sphere geometry therefore has geometric and representational deficiencies. By examining how MPM often initializes the particles in a particle model, we conclude that *cuboid* glyphs produce meaningful and insightful visualizations. Furthermore, we take advantage of the generality inherent in MPM's modeling algorithm to create *hexahedral* particle models that do not suffer from the axis-alignment constraint placed upon cuboid particle models.

3.1 Cuboid Glyphs

When initializing a particle model, GIMP requires that each particle have a position and a volume (which is used to normalize integrals over other particle values). GIMP has an implicit notion of the shapes of particles (as described below, and in Figure 1), and we argue it is useful for visualization, in addition to computation.

A standard way to create an MPM model regularly divides the background grid cells into subcells (Figure 1). If the centroid of a subcell falls inside the boundary of the object being modeled, then a particle is initialized at the centroid location, with volume equal to the volume of the subcell. Because the initial volume is derived from the subcell's shape, the modeling process further implies that each particle actually looks like a cuboid with the same dimensions as the subcell. In other words, a *cuboid* glyph geometry is suitable for visualizing this model. Using cuboids in this case produces a visually continuous model occupying exactly the volume the modeling process dictates. The particles now look like the continua of matter they represent, connected at the faces in such a way that the particle glyphs reflect visually the manner in which the object is partitioned numerically.

Cuboids are used primarily because they reflect the underlying numerical representation of the material, but they also have desirable graphical properties that enhance the visualization. Cuboids are bounded by quadrilateral faces, enabling more effective visualization of deforming surfaces. For instance, the exposed faces of cuboids on the outer edge of an object represent its surface; as the object deforms, lighting effects help the viewer visually track the surface as it changes shape. Put another way, sphere glyphs allow a viewer to track a deforming surface, but only by their relative positions; cuboids, on the other hand, show relative positions *and* relevant lighting cues to give a much stronger sense of a deforming surface. Furthermore, in a cuboid glyph model, the edges of the cuboids form a kind of grid of junction lines that is visible in visualizations. As the simulation progresses, these junction lines change their shape to reflect the changing shape of the particles (Figure 2); they can also be used to track deformation on a larger scale (Figure 3).

3.2 Hexahedral Glyphs

Because the MPM background mesh is most often a rectilinear grid, the boundary of the particle models produced by the standard modeling algorithm are also constrained to be rectilinear. Models made to approximate objects with curved boundaries will have a stairstep or lego-brick quality along these curves (Figure 1).

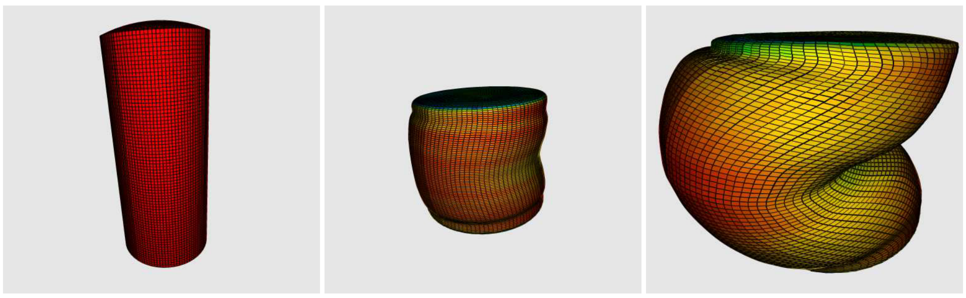
Figure 2: A cylinder, discretized in CUBIT, is compressed by rigid plates (not shown) until it buckles, introducing a large deformation.

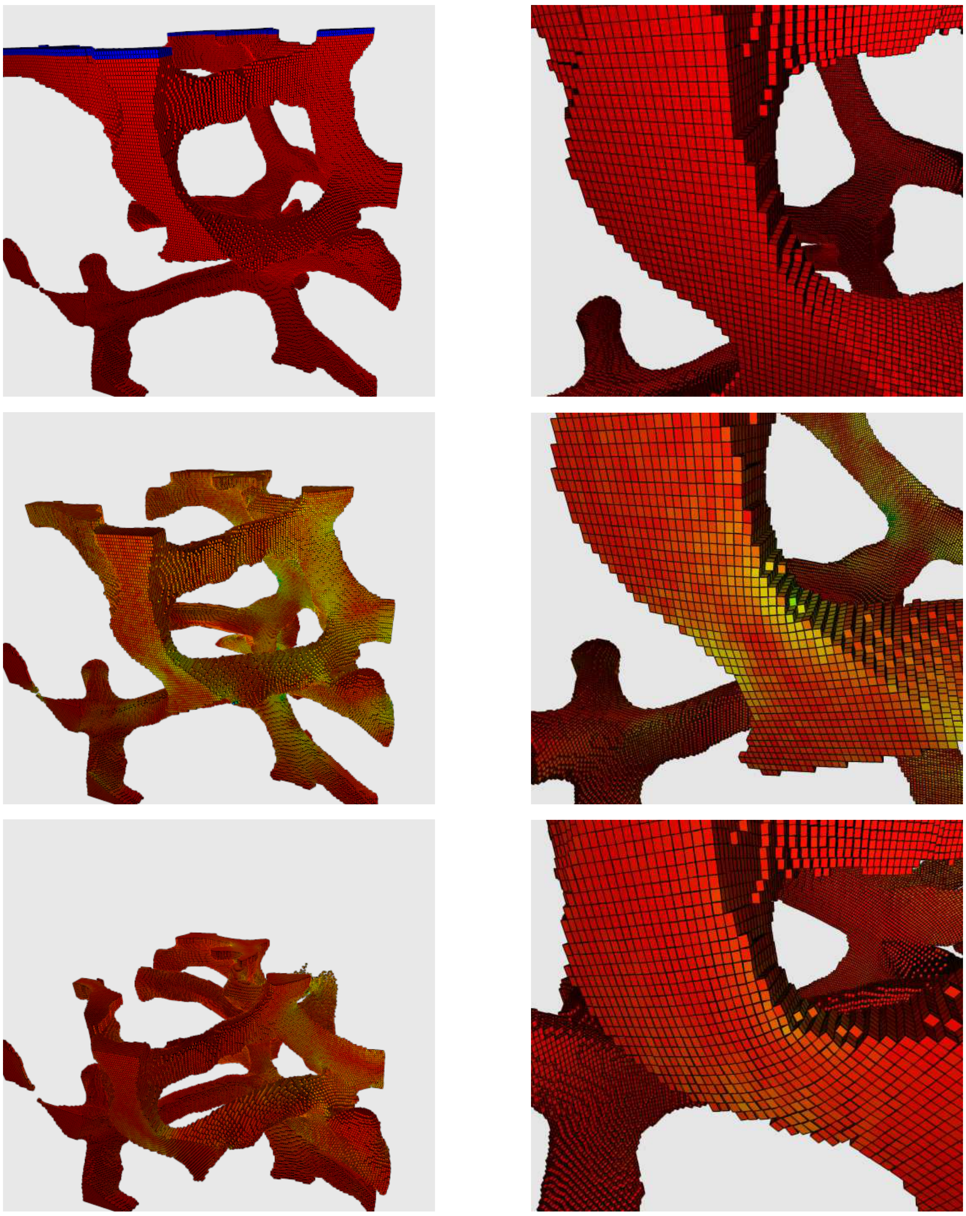

Figure 3: Open-celled foam made up of microstruts is compacted by a rigid plate moving downward (not shown). The struts deform in bearing the load as the total volume of the foam decreases. The strut in the left of the foreground deforms considerably during the process (detail, right column). This strut is originally vertical and bends during compression. In the lower right image the rotation associated with this deformation can be observed by tracking the boundaries between particles. Along the bottom edge of the strut, where tensile stresses are acting, some particle separation can be seen, possibly reflecting some numerical inaccuracies in the simulation.

Initially, the particle glyphs will be axis-aligned; instead of seeing the surface of a sphere, for example, we will only see a stairstep approximation to that surface. By generalizing cuboids to *hexahedra*, however, we can remove this restriction on the initial orientation of the particle glyphs.

CUBIT [Sandia National Laboratories (2007)] is a software package that can produce hexahedral meshes.[2] Such meshes have several desirable properties for MPM particle models: they approximate the interior of some boundary, covering it continuously with volumetric elements, and the exposed faces of the boundary mesh elements can be made to approximate a chosen surface. To create a particle model from a hex mesh, we simply initialize one particle at the geometric centroid of each mesh element, with its initial volume set to that of the mesh element. This list of material points is given to the MPM algorithm, which can then carry out a simulation. CUBIT can create meshes to represent cuboids, cylinders, prisms, cones, polygonal pyramids, spheres, and toruses, and it can compose these primitives into more complex shapes using constructive solid geometry. It can therefore offer a very general range of particle models. To perform the visualization, each particle is represented by a hexahedral glyph with the same shape as the mesh element that produced it and then transformed about its centroid by the deformation gradient tensor. This approach allows the simulation scientist to create more realistic-looking models, which may lead to better insight gained from visualization.

4 Examples and Discussion

To test our method, we have run several MPM simulations with different geometry and physical conditions. Some of the simulations aim to produce specific modes of deformation for observation, while others are real data produced by the C-SAFE project [Henderson, McMurtry, Smith, Voth, Wight, and Pershing (2000)]. The images were all produced by the Manta ray tracer [Bigler, Stephens, and Parker (2006)], which includes spheres, cuboids, and hexahedra as graphical primitives, and runs at interactive frame rates on modest desktop hardware. We used a variation of the standard ray tracing algorithm to also render non-photorealistic intersection, crease, and silhouette lines [Choudhury and Parker (2009)]; these lines help to show the spatial relationships of the individual glyphs.

Figure 3 gives an example of our method applied to real data. The simulation shown involves a small volume of foam, whose microstruts are visible, being crushed by a downward-moving rigid plate. The geometry for the foam model was created by obtaining X-ray microtomography data of a real foam sample, then using imaging

[2] Of the ten topological classes of hexahedra, this paper uses "hexahedron" to mean "quadrilateral-faced hexahedra," the variety topologically equivalent to a cube.

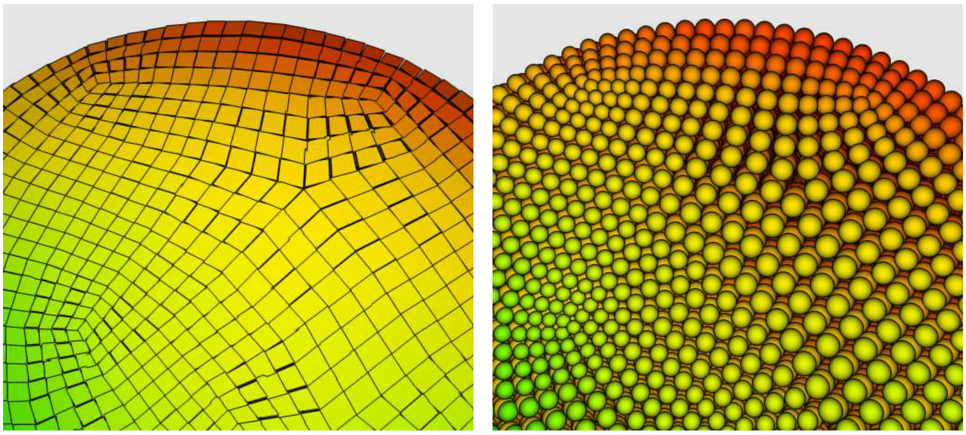
Figure 4: Bottom of cylinder depicted in Figure 2. Hexahedral glyphs emphasize the volumetric nature of MPM particles; spheres emphasize their pointwise nature. In particular, slight particle separation is visible in the left image; such separations are not apparent in the right image.

techniques on the resulting data volume and initializing particles for voxels surpassing a threshold intensity (indicating the presence of foam in that voxel) [Brydon, Bardenhagen, Miller, and Seidler (2005)]. As the particles in this case were derived from voxel data, we use cuboid glyphs for visualization. The right column of Figure 3 shows a close-up of one strut that deforms considerably during the simulation. This figure and all others in this paper use equivalent stress to colormap the particles. We also note here that in the interest of focusing on the deformation rather than scalar values, we have chosen to omit colorscale legends in these figures as well.

4.1 Physical Basis

The primary feature of our visualizations is that they are physically based, determining glyph shape from the modeling process, and then deforming the glyphs in accordance with deformation data generated during the simulation. In particular, the method provides a *volumetric view* of the particles, showing how they would appear under the assumption that the entire particle voxel deforms with a constant deformation gradient, as computed at the particle's nominal position. Figure 4 shows the bottom surface of a cylinder that is being being crushed longitudinally by two rigid plates (as shown in Figure 2). The particles in this part of the cylinder are experiencing loads that tend to flatten them out. The hexahedral glyphs in Figure 4, left, are therefore flat and thin, so that they occupy more screen space

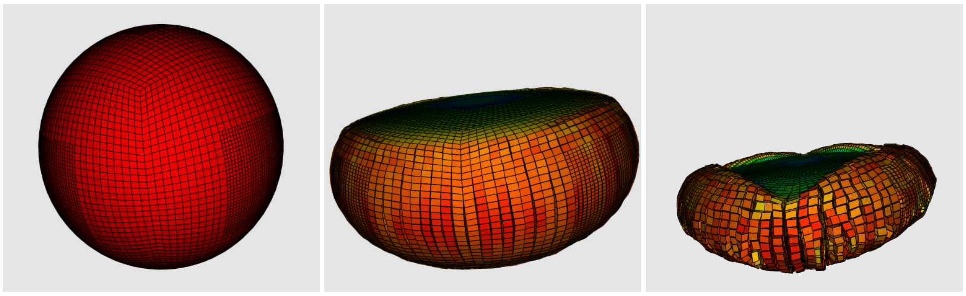

Figure 5: A sphere is compressed by rigid plates (not shown) until it is flat in the middle. Some material oozes out from between the plates, spreading out in such a way that the particles become separated. Because the material model did not include a failure model in this simulation, the particle separation indicates the approximate nature of the simulation. The larger separations in the rightmost image suggest possible error as well.

than their spherical counterparts in the right image. On the other hand, in Figure 4, right, one can actually see past the spheres into the interior of the model, constituting a *computational view* of the particles that emphasizes their pointwise nature, as they exist during computation by the MPM algorithm. By using small, volume-independent spherical glyphs, we can observe the relative positions of particles throughout the volume of a simulated body (for this purpose, global illumination can also help [Gribble, Stephens, Guilkey, and Parker (2006); Gribble and Parker (2006); Tarini, Cignoni, and Montani (2006)]).

The major strength of hexahedral glyphs is that they give a good physical picture of what the simulated object might look like in reality, which also has other implications. For example, MPM simulates continuum bodies, which means that material separation does not occur unless a material failure model is included. The particles in MPM are not connected as in a finite element mesh, thus, the degree to which edges of cuboid particles remain connected visually is a reflection that the method is accurately capturing the deformation that an object experiences. However, as the simulation is necessarily an approximation to the true behavior, we may observe separation between particles in the resulting visualization. Such visual separation reflects both the approximate nature of the simulation results, as well as the visualization assumption that the whole particle deforms according to a constant per-particle value of **F** (Figure 4, left). However, large, localized separations can indicate error in the simulation (Figure 5).

Such separations are not apparent when using spherical glyphs because spheres do

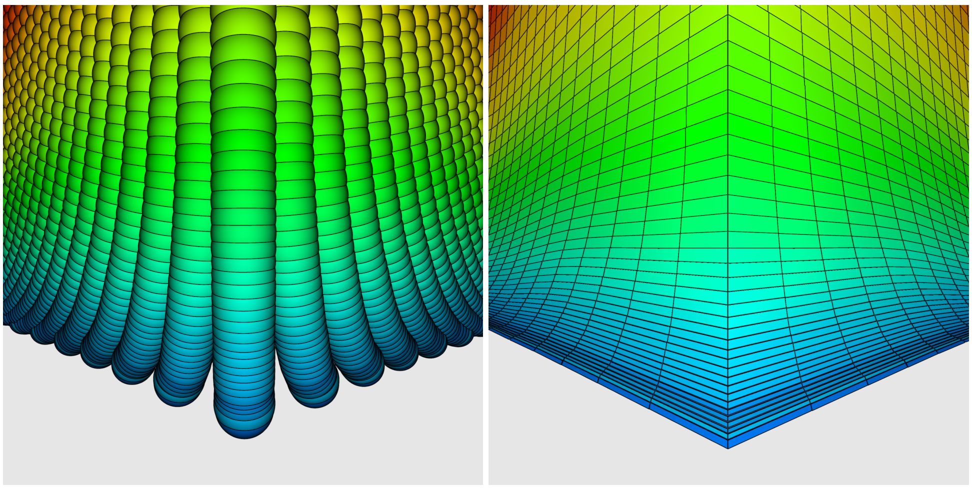

Figure 6: Extreme deformation is induced by very high compressive stresses in the center of a Taylor-impacted cylinder. By changing their shape to be flat and wide, cuboid glyphs are able to illustrate the deformation; by comparison, the sphere glyphs become impacted, hiding each other from view, and imply that the columns of particles are becoming separated.

not properly model the material continuum. This is a case of the visualization drawing attention to errors in the simulation, providing hints to the simulation scientist about how the simulation quality might be improved. In Figure 5, right, the separation is quite large and may, for example, indicate the need to run the simulation again for higher accuracy, using a discretization with a larger number of smaller particles in that area.

Several of the simulations demonstrate the usefulness of using a volumetric view. Figure 6 shows the results of a Taylor impact [Taylor (1948)] simulation, in which a metal cylinder is shot onto a rigid surface and deforms. Because this setup is radially symmetric about the cylinder's axis, the simulation uses only one quarter of the cylinder with appropriate boundary conditions. Looking at the internal corner of the model actually shows what is happening in the center of the cylinder. The extremely high compressive stress in this region induces the particles to become very wide and thin. In the visualization, the particles remain continuous, suggesting the numerical stability of the simulation.

Figure 7 shows a rubber sheet with its edges affixed to a wall being struck by a projectile from underneath. The middle part of the sheet moves upward with the projectile, and then returns to its original position while an elastic wave travels

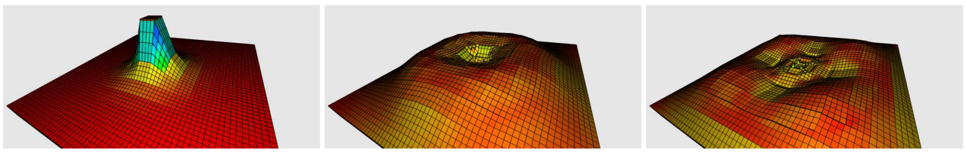
Figure 7: A rubber sheet fixed around its edges to a wall, struck by a projectile from underneath. An elastic wave travels outward from the point of impact.

out from the point of impact. The graphical qualities of the cuboid glyphs nicely illustrate the curve of the wave as it travels. The wave travels outward radially, but the front strikes different parts of the boundary at different times, taking longer to reach the corners of the sheet than the sides. The reflected waves therefore return at different times and cause a characteristic interference pattern, as illustrated by the shading in the rightmost image of Figure 7.

Furthermore, this particular visualization reveals something interesting about the MPM algorithm itself. It is subtly apparent in Figure 7 that the particles are somehow organized into "tiles" with similar surface orientation. These are visible from the changes in shading resulting from the slight change in the angles of the tiles with respect to each other. The effect results from the use of the background grid used in MPM, which gathers and then reprojects changes to particle state through basis functions during the simulated timestep. This is an example of the visualization revealing a quality of the algorithm itself, in much the same way as small gaps appearing between particles reflect the approximating nature of any simulation algorithm.

4.2 Visual Cues for Geometric Features

When studying deformation in simulation data, it is important that the visualization not misrepresent or obscure important features. Hexahedral glyphs are a way to eliminate such misrepresentations, as demonstrated in the bottom corner of the cylinder (Figure 8). In the sphere scene, the cylinder's corner seems to occur along the row of green particles three layers above the blue ones, but the hex scene demonstrates that this is not the case. In fact, the corner appears in the row of blue particles, as evidenced by the two faces visible in that row. What looks like the corner in the sphere scene is in fact a bulge that occurs just above the plane of contact with the rigid plate. The shading and orientation of the hex faces shows this phenomenon quite clearly.

When viewing a dataset at close ranges to investigate small-scale features, global indicators of structure and deformation are missing. The inside of the buckle feature

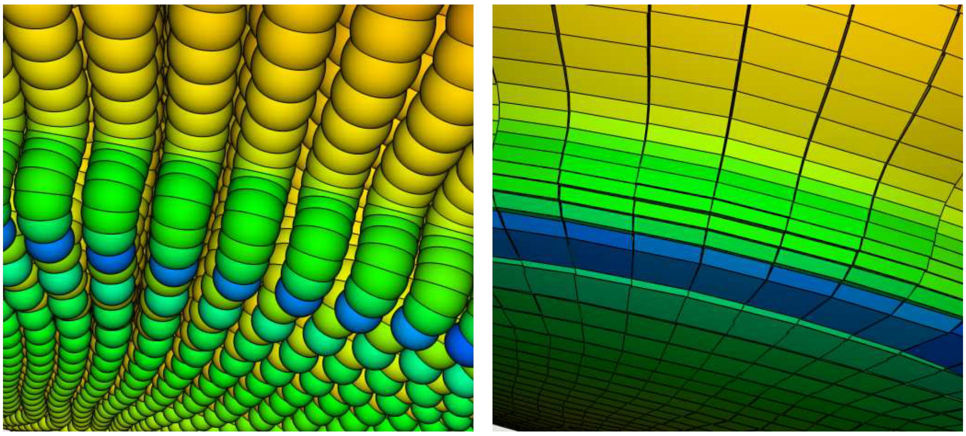
Figure 8: The spheres falsely imply that the corner of the cylinder lies along the green particles, while the hexahedra show that actually, the corner occurs in the blue particles, and the green particles show a slight bulge just above it.

in the crushed cylinder (Figure 9) demonstrates this problem. A sphere's surface normal varies continuously across its surface; many spheres packed close together in a visualization will therefore look very similar. For this reason, in the left image of Figure 9 it is very difficult to infer the actual distribution of the glyphs. It is not clear if the curved bands of connected spheres lie in the same plane, or if they recede from the viewpoint. The structure is much easier to see in the right image, because the exposed faces of the hexahedral glyphs have one normal vector each, and thus can serve as area elements making up a surface, even at close viewing distances.

5 Conclusions and Future Work

We have demonstrated an extension to glyph-based particle visualization methods that includes the deformation gradient and therefore visualizes deformation directly. Our major strategy has been to understand a given particle method's approach to modeling the geometry of an individual particle, and then adopt that geometry as the visualization glyph. This strategy moves particle visualization away from arbitrary choices, such as spheres, that seem to be appropriate, but turn out to be deficient upon close inspection.

By visualizing the particles as the hexahedral regions they represent rather than abstract points in space, scientists can get a clearer and more direct understanding of how a simulation affects the material being modeled. This directness is espe-

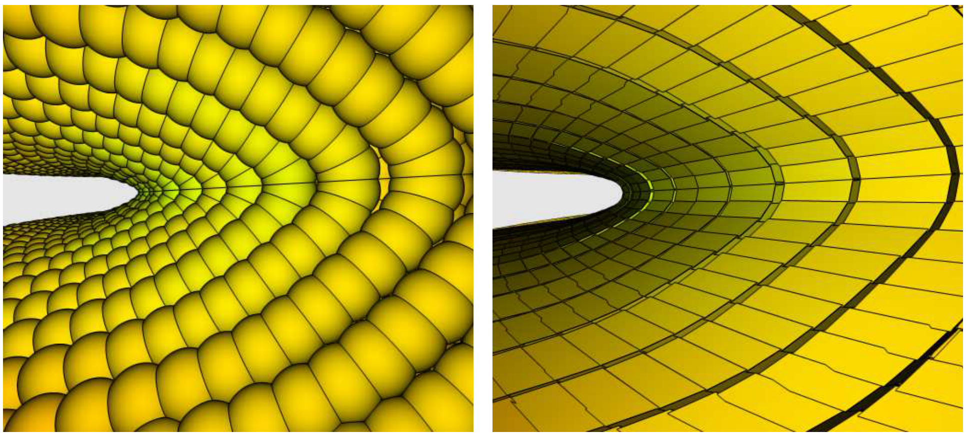
Figure 9: Hexahedra show the geometry of high curvature areas more clearly than spheres can through shading cues that suggest surfaces directly.

cially important when deformation is the central object of interest, as in the study of material failure during a catastrophic explosion, or stress testing of structures. Furthermore, including deformation data directly in the visualization allows simulation scientists to detect errors in their simulations earlier, as when the visualization shows deformations that clearly do not reflect reality.

In future work we wish to integrate other common visualization techniques. For instance, it has been demonstrated that using ambient occlusion or other approximations to global illumination can enhance user perception of particle datasets [Gribble and Parker (2006); Tarini, Cignoni, and Montani (2006)]; presumably this is true when using hexahedra rather than spheres as well.

Acknowledgement: This work was supported by the U.S. Department of Energy through the Center for the Simulation of Accidental Fires and Explosions, under grant W-7405-ENG-48; and by the NIH/NCRR Center for Integrative Biomedical Computing, under grant P41-RR12553-10.

The authors would like to thank Jason Shepherd for discussions about hex meshing, and his help with the CUBIT software package.

References

Atluri, S. N.; Liu, H. T.; Han, Z. D. (1998): A new meshless local petrov-galerkin (mlpg) approach in computational mechanics. *Computational Mechanics*, vol. 22, no. 2, pp. 117–127.

Bardenhagen, S. G.; Kober, E. M. (2004): The generalized interpolation material point method. *CMES: Computer Modeling in Engineering and Sciences*, vol. 5, no. 6, pp. 477–496.

Bathe, K. (1996): *Finite Element Procedures*. Prentice-Hall, New Jersey.

Belytschko, T.; Krongauz, Y.; Organ, D.; Fleming, M.; Krysl, P. (1996): Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, vol. 139, no. 1-4, pp. 3–47.

Bigler, J.; Guilkey, J.; Gribble, C. P.; Hansen, C. D.; Parker, S. G. (2006): A case study: Visualizing material point method data. In *Proceedings of Euro Vis 2006*, pp. 299–306, 377.

Bigler, J.; Stephens, A.; Parker, S. (2006): Design for parallel interactive ray tracing systems. In *Proceedings of The IEEE Symposium on Interactive Ray Tracing*, pp. 187–196.

Brackbill, J. U.; Ruppel, H. M. (1986): Flip: A method for adaptively zoned, particle-in-cell calculations of fluid in two dimensions. *Journal of Computational Physics*, vol. 65, no. 2, pp. 314–343.

Brydon, A.; Bardenhagen, S.; Miller, E.; Seidler, G. (2005): Simulation of the densification of real open-celled foam microstructures. *Journal of the Mechanics and Physics of Solids*, vol. 53, no. 12, pp. 2638–2660.

Choudhury, A. I.; Parker, S. G. (2009): Ray tracing NPR-style feature lines. In *NPAR '09: Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pp. 5–14. ACM.

Gribble, C.; Stephens, A.; Guilkey, J.; Parker, S. (2006): Visualizing particle-based simulation datasets on the desktop. In *Proceedings of the British HCI 2006 Workshop on Combining Visualization and Interaction to Facilitate Scientific Exploration and Discovery*.

Gribble, C. P.; Parker, S. G. (2006): Enhancing interactive particle visualization with advanced shading models. In *APGV '06: Proceedings of the 3rd symposium on Applied perception in graphics and visualization*, pp. 111–118, New York, NY, USA. ACM Press.

Guilkey, J. E.; Harman, T. B.; Banerjee, B. (2007): An eulerian-lagrangian approach for simulating explosions of energetic devices. *Comput. Struct.*, vol. 85, no. 11-14, pp. 660–674.

Guilkey, J. E.; Weiss, J. A. (2003): Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering*, vol. 57, pp. 1323–1338.

Hall, F. R.; Hayhurst, D. R. (1991): Continuum damage mechanics modelling of high temperature deformation and failure in a pipe weldment. *Proceedings: Mathematical and Physical Sciences*, vol. 433, no. 1888, pp. 383–403.

Harlow, F. H. (1964): The particle-in-cell computing method for fluid dynamics. *Methods of Computational Physics*, vol. 3, pp. 319–343.

Henderson, T.; McMurtry, P.; Smith, P.; Voth, G.; Wight, C.; Pershing, D. (2000): Simulating accidental fires and explosions. *Computing in Science and Engineering*, vol. 2, no. 2, pp. 64–76.

Hoover, W. G. (2006): *Smooth Particle Applied Mechanics: The State of the Art*, volume 25 of *Advanced Series in Nonlinear Dynamics.* World Scientific.

Ionescu, I.; Guilkey, J.; Berzins, M.; Kirby, R.; Weiss, J. (2006): Simulation of soft tissue failure using the material point method. *Journal of Biomechanical Engineering*, vol. 128, no. 6, pp. 917–924.

Irving, G.; Teran, J.; Fedkiw, R. (2004): Invertible finite elements for robust simulation of large deformation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 131–140, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Kindlmann, G. (2004): Superquadric tensor glyphs. In *Proceedings of IEEE TVCG/EG Symposium on Visualization 2004*, pp. 147–154.

Liu, G.-R. (2003): *Mesh Free Methods: Moving Beyond the Finite Element Method.* CRC Press.

Love, E.; Sulsky, D. L. (2006): An unconditionally stable, energy-mpmentum consistent implementation of the material-point-method. *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 33-36, pp. 3903–3925.

Maloney, C.; Lemaître, A. (2004): Universal breakdown of elasticity at the onset of material failure. *Physical Review Letters*, vol. 93, no. 195501.

Meakin, P. (1991): Models for Material Failure and Deformation. *Science*, vol. 252, pp. 226–234.

Monaghan, J. J. (2005): Smoothed particle hydrodynamics. *Reports on Progress in Physics*, vol. 68, no. 8, pp. 1703–1759.

O'Brien, J. F.; Bargteil, A. W.; Hodgins, J. K. (2002): Graphical modeling and animation of ductile fracture. *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 291–294.

O'Brien, J. F.; Hodgins, J. K. (1999): Graphical modeling and animation of brittle fracture. In *SIGGRAPH '99: Proceedings of the 26th annual conference on*

Computer graphics and interactive techniques, pp. 137–146, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Parker, S.; Johnson, C. (1995): SCIRun: A scientific programming environment for computational steering. In *Supercomputing '95*. IEEE Press.

Sandia National Laboratories (2007): The CUBIT geometry and mesh generation toolkit. http://cubit.sandia.gov, 2007.

Steffen, M.; Wallstedt, P.; Guilkey, J.; Kirby, R. M.; Berzins, M. (2008): Examination and analysis of implementation choices within the material point method (MPM). *Computational Modeling in Engineering and Science*, vol. 31, no. 2, pp. 107–127.

Strang, G. (1988): *Linear Algebra and its Applications*. Harcourt Brace Jovanovich College Publishers, third edition.

Sulsky, D.; Chen, Z.; Shreyer, H. (1994): A particle method for history dependent materials. *Computer Methods in Applied Mechanics and Engineering*, vol. 118, pp. 179–196.

Sulsky, D.; Zhou, S.-J.; Schreyer, H. L. (1995): Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, vol. 87, no. 1–2, pp. 236–252.

Tarini, M.; Cignoni, P.; Montani, C. (2006): Ambient occlusion and edge cueing for enhancing real time molecular visualization. volume 12, pp. 1237–1244, Los Alamitos, CA, USA. IEEE Computer Society.

Taylor, G. (1948): The use of flat-ended projectiles for determining dynamic yield stress, part i. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 194, no. 1038, pp. 289–299.

Terzopoulos, D.; Fleischer, K. (1988): Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pp. 269–278, New York, NY, USA. ACM Press.

Terzopoulos, D.; Platt, J.; Barr, A.; Fleischer, K. (1987): Elastically deformable models. *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 205–214.

Wallstedt, P. C.; Guilkey, J. E. (2008): An evaluation of explicit time integration schemes for use with the generalized interpolation material point method. *Journal of Computational Physics*, vol. 227, no. 22, pp. 9628–9642.