Eurographics/ IEEE-VGTC Symposium on Visualization (2006)
Thomas Ertl, Ken Joy, and Beatriz Santos (Editors)

A Case Study: Visualizing Material Point Method Data

James Bigler[1], James Guilkey[2], Christiaan Gribble[1], Charles Hansen[1] and Steven Parker[1]

[1]Scientific Computing and Imaging Institute, University of Utah, USA
[2]Dept. of Mechanical Engineering, University of Utah, USA

Abstract

The Material Point Method is used for complex simulation of solid materials represented using many individual particles. Visualizing such data using existing polygonal or volumetric methods does not accurately encapsulate both the particle and macroscopic properties of the data. In this case study we present various methods used to visualize the particle data as spheres and explain and evaluate two methods of augmenting the visualization using silhouette edges and advanced illumination such as ambient occlusion. We also present informal feedback received from the application scientists who use these methods in their workflow.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.3.3 [Computer Graphics]: Display algorithms I.3.3 [Computer Graphics]: Bitmap and framebuffer operations I.4.6 [Image Processing]: Edge and feature detection

1. Introduction

Computational simulation is frequently done on grids; however, grid based solution methods, such as the finite element method, are not always well suited for complicated geometry where construction of a suitable grid is unreasonably time consuming. Additionally, some such methods are not robust when materials are subject to large deformations. So called “meshless" or “quasi-meshless" methods are an increasingly popular alternative for certain classes of problems. In meshless methods, structures are generally represented using particles, each of which is a discrete representation of a small region of the structure.

One particular particle based method is the Material Point Method (MPM) [SCS94, SZS95, GW01]. MPM facilitates representation of arbitrarily complex geometries, is robust with respect to large deformations, and has advantages over strictly grid based methods in simulations involving contact between multiple objects [BBS00]. This method also introduces new visualization challenges. The scientists need to see both the general shape of the object as well as fine structure such as developing cracks. They also need to be able to visualize time varying data interactively in order to investigate features of the simulation as it progresses. The merits of applying ambient occlusion shading and silhouette edging were investigated via informal feedback from the application scientists who regularly visualize MPM data sets.

2. The Material Point Method

The Material Point Method (MPM) is a particle method for structural mechanics simulations used to evolve the state of solid materials. Lagrangian particles, or material points, are used to discretize the volume of a material; each particle carries state information about the portion of the volume that it represents, including mass, volume, velocity, stress, etc. A regular structured grid is used as a computational scratchpad for the integration and solution of the weak form of the equations of motion. A typical region of the computational domain is depicted in Figure 1, panel 1, where the domain of interest is discretized into particles and a regular Cartesian grid is created over the entire domain. The physical state of the particles is projected to the grid nodes using a suitable shape function, (Figure 1, panel 2). Stress is computed at the particles based on the gradient of velocity at the grid nodes, and the equations of motion are subsequently solved on those nodes. This solution may be carried out using either explicit [SCS94] or implicit [GW03] time-integration. In either case, the results of the time integration are increments in the position, velocity and temperature of the nodes (Figure 1, panel 3). Those increments are interpolated to the particles and their physical state is updated (Figure 1, panel 4).

Traditionally, at this point, the deformed computational grid is reset to its original undeformed configuration (Figure 1, panel 5a) and the simulation proceeds to the subsequent time step. However, for certain types of simulations, it may be advantageous not to reset the mesh at the end of a time step (Figure 1, panel 5b). Even when this option is used, the grid can be reset at any time to the undeformed configuration, without the difficulties of remeshing that would be required when using the Finite Element Method, since the data reside at the particles.

3. Visualization Support for MPM

The unique qualities of MPM data make it clear that there exists a need to view particle data in a way that is both appropriate and informative. What does this mean? In MPM, particle data seeks to represent pieces of a larger whole, and the ability to see and interpret the macroscopic structure created by these particles is vital. As well, the ability to view fine structure is important for the visualization of MPM data. Appropriate visualizations are those that add to the overall understanding of the data while minimizing erroneous interpretations of it. There are two main goals of MPM data visualization. The first is the structure that is represented by the actual placement and sizes of the particles. The second are qualitative trends in values associated with the particles such as mass, speed, or stress.

Because MPM makes use of a grid as a computational scratchpad, the output produced by the simulations contains both particle and regular grid data. One way to view the results of MPM simulations is to render the particle values interpolated onto a regular grid using traditional grid based methods such as isosurfacing [LC87] and direct volume rendering [Lev88]. However this has several disadvantages. Areas where the particles are sparse can result in incorrect structure reconstruction, missing features too fine for the chosen grid resolution. To compensate for this, grids must be refined beyond the resolution of the particles. Also, memory is wasted representing areas that do not need refinement, such as regions without data or where the values vary smoothly. Representing particle data as a regular grid may also mask the fine structure in the original form. For these reasons and user preference, it is advantageous to visualize the particles.

Particle visualization often takes the form of rendering the data as spheres or ellipsoids to represent the location and size of the particles used in simulation [KPH96, Gum03, BKHJ01, KBHJ01]. Color mapping scalar quantities associated with particles (such as mass, volume, and speed) is an additional method used to obtain a qualitative understanding of the data. Other methods of applying particle values to the dataset include particle deformation and rotation resulting in ellipsoid or super-ellipsoid primitives (Figure 2).

The SCIRun problem solving environment is the primary visualization system used by MPM computational scientists for debugging and initial inspections of smaller data

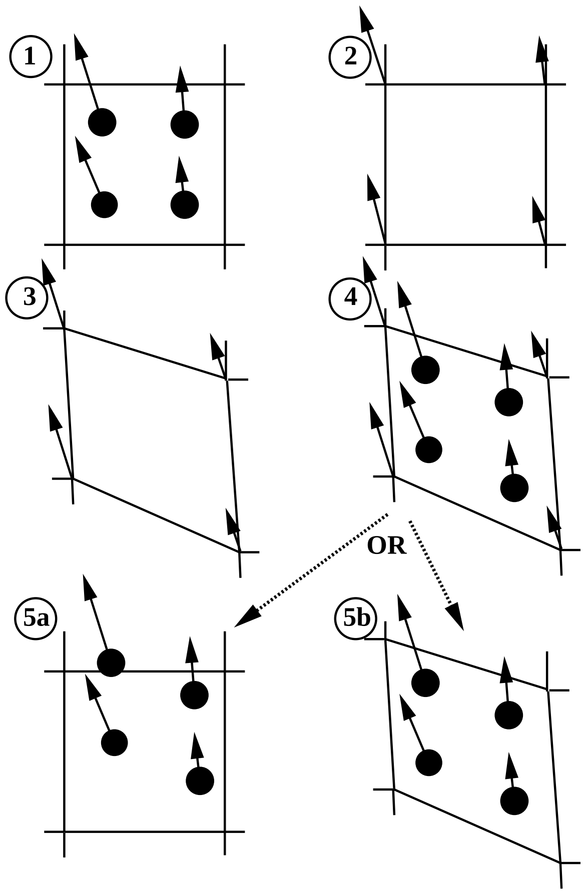

Figure 1: *Illustration of the steps in the MPM algorithm for particles occupying a single cell of the background grid. 1. A representation of four material points (filled circles), overlaid with the computational grid (solid lines). Arrows represent displacement vectors. 2. The material point state vector (mass, volume, velocity, etc.) is projected to the nodes of the computational grid. 3. The discrete form of the equations of motion is solved on the computational grid, resulting in updated nodal velocities and positions. 4. The updated nodal kinematics are interpolated back to the material points and their state updated. 5a. In the standard MPM algorithm, the computational grid is reset to its original configuration and the process repeated. 5b. In a modification to the algorithm, the grid is not reset, but is allowed to move with the particles, thereby retaining the optimal distribution of particles with respect to the grid.*

sets [WPS*05, JPH*99]. SCIRun provides the opportunity to view the particle data as spheres or ellipsoids, or as isosurfaces, using values interpolated to a grid and desktop graphics hardware (see Figure 2). Larger simulations produce hundreds of thousands to millions of particles. Current graphics hardware is incapable of rendering these large numbers of spheres in real time. To produce interactive displays of results it is necessary to use ray tracing on large parallel machines, such as a SGI Origin system. This is the primary method utilized by scientists performing computational experiments with MPM. Ray tracing also provides the opportunity to view multiple time steps with shadows in an interactive setting.

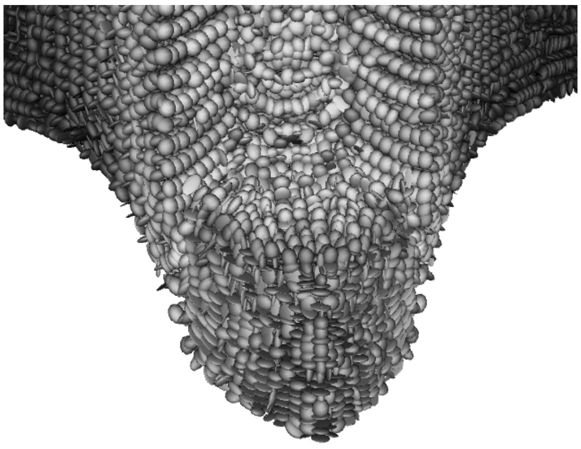

Figure 2: *Tensor values can be used to deform and rotate spheres representing particles into ellipsoids. Color version (Figure 12).*

Local lighting models that use the surface normal to compute diffuse and specular components are often employed when rendering and provide reasonable surface shape information. Since MPM geometry is made entirely of spherical primitives, this illumination model is insufficient for determining macro shape of objects. Local lighting models also fail to aid in the perception of the spatial relationships of particles with each other. This global information cannot be accurately captured with a local model. Using a global operation, such as shadows, can provide information about proximity of structures and help disambiguate relative positions of the geometry [WFG92, Pal99]. Application scientists visualizing their data often employ shadows to aid viewing the global structure of their data. Shadows, however, create regions where the ambient term of the shading model becomes dominate. With many local shading models, the ambient term becomes constant in regions of shadow or where the surface faces away from the light source, reducing the spatial cues needed for particle visualization. Consequently, non-constant ambient values that take into consideration global positioning become important. Stewart's work on ambient occlusion (which he called vicinity shading) for isosurface volumes provided inspiration for the application of this technique for particle rendering [Ste03]. Results could be combined with arbitrary direct illumination, because the obscurance computation was a property of the geometry and not of the lighting conditions. The method was also useful for interactive applications, because the results were view independent [ZIK98, IKSZ03, Lan02].

Illumination can help with shape perception, but silhouette edges can be used to accentuate a set of view dependent boundaries [GGSC98, GSG*99, Goo98]. These edges are usually rendered in black and follow the view-dependent hull of the object. This helps define macroscopic structure of the geometry. Silhouette edge computation usually falls in three categories: image based, object based, and hybrids of the two.

There have been several hybrid and object based methods that use both object and image based techniques [RC99, BS00, GGSC98, GSG*99, NM00, KWTM03]. These methods, however, are targeted towards polygonal based geometry and attempt to extract all the silhouette edges. Also we wished to produce silhouette edges not for all spheres but across groups of spheres that had no underlying connectivity aside from their relative proximity. We therefore employ on an image based method derived from the ones used by Saito and Takahashi [ST90] and McCool [McC01]. Both of these methods used the depth buffer to produce silhouette edges for objects in the scene without respect for the geometry in the scene. Convolving the depth buffer with derivative kernels provided a simple method of determining discontinuities in the geometry.

The ability to render millions of particles in an interactive setting that allows for visualization of the global structure without compromising the ability to inspect fine features is necessary for MPM data visualization. Using an interactive ray tracer similar to the one developed by Parker et al. [PMS*99, PPL*99], scientists are able to interactively view time varying visualizations of particle data sets that make use of local lighting in addition to shadows. Responding to the need for better particle visualizations, methods of visualizing particle sets using ambient occlusion and silhouette edges were developed.

3.1. Ambient Occlusion

Because it is important to have interactivity while exploring the MPM data sets, any advanced illumination techniques used would need to minimize impact on interactive frame rates. Since the cost of computing ambient occlusion would be too costly during rendering, we decided to precompute these values for later use during rendering. Ambient occlusion was utilized over full global illumination, because ambient occlusion is a measure of the geometry and allowed for changes in the lighting during rendering.

Ambient occlusion values are precomputed as textures that were then mapped to particles during the rendering phase using a simple UV globe mapping. During precomputation the lighting for each texel is determined using stratified sampling. Bilinear interpolation reconstructs the value on the sphere during rendering. In order to compensate for leakage of values from occluded areas into visible ones, the degree of occlusion for each texel is stored and values of visible texels were then bled into occluded areas. This reduced the regions of black bands visible between intersecting spheres.

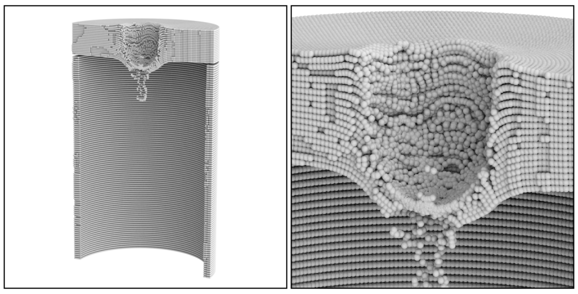

Figure 3: *Ambient occlusion shading texture values mapped on an aluminum cylinder impacted by a steel sphere (not shown). This configuration has been tested experimentally and provides a source of data for material model validation. One thing to note is the ability to see the two spheres standing out in front of the hole. Subimages are referred to as A (left) and B (right). It took 25 minutes to precompute the textures. Frame rates to render image A (500x500) using direct lighting only, direct lighting with ambient occlusion, and ambient occlusion only were 15.22 f/s, 14.04 f/s, and 15.14 f/s respectively. Frame rates to render image B (500x500) were 8.19 f/s, 7.84 f/s, and 8.35 f/s.*

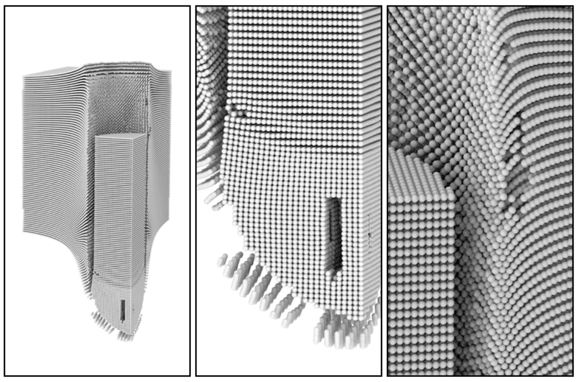

Figure 4: *Ambient occlusion shading texture values mapped on a steel projectile impacting an iron target. Extensive data has been collected on this configuration, making it a valuable tool for validating material and contact models. Subimages are referred to as A, B, and C, left to right. Figure A is provides a view of the whole data set, while B and C show some close ups. The relative positions of the bullet with the material as well as inside the holes in the objects are easily perceived. It took 33 minutes to precompute the textures. Frame rates to render image A (250x500) using direct lighting only, direct lighting with ambient occlusion, and ambient occlusion only were 32.89 f/s, 28.75 f/s, and 30.50 f/s respectively. Frame rates to render image C (250x500) were 24.62 f/s, 22.66 f/s, and 26.32 f/s.*

The affect on rendering times using both direct lighting and texturing the ambient occlusion values was about a 10% drop in frame rates using 20 processors on an SGI Origin 3800 with 600 MHz R14K processors. The textures provide some variance of lighting over the surface, and can be used without the addition of the direct lighting. Figure 8 shows a color mapped data set with and without the addition of direct lighting. For many cases using only the ambient occlusion values can provide frame rates nearly as good as or better than using only direct lighting.

The precomputation times were largely dependent on the size and complexity of the geometry. Each sphere was textured with 256 texels (16x16), and each texel was computed with 49 samples. Times to precompute the textures ranged from 25 minutes to 12 hours depending on the complexity and size of the geometry. See Figures 3, 4, 5, 6, and 7 for times to precompute the data on 20 processors of an SGI Origin 3800 with 600 MHz R14K CPUs. It should be noted that these times were for only single time steps. To precompute large time sequences can take hours of CPU time. Since all of the textures we used were 16x16 (or 256 texels), 256 additional bytes must be stored for each particle. The memory to store and time to precompute the textures were found to be prohibitive for the large time sequences used by the application scientists.

3.2. Silhouette Edge Enhancement

A popular method of computing edges in images is using the Laplacian of a Gaussian (LoG). This kernel is convolved with the image and edges are marked where there are zero crossings. Convolution of the Gaussian is unnecessary in our case, because the depth buffer is already relatively smooth. Therefore, only the Laplacian is needed to look for zero crossings in the second derivative. It is done using a 3x3 kernel.

Zero crossings are discovered by taking the difference between the center sample and the remaining surrounding samples. Since all depth values are positive, nonzero differences indicate a zero crossing or an edge. One thing to take into consideration is double edges produced where there are discontinuities in the first derivative. To prevent these double lines, only differences that are positive are included.

If all the edges are detected our image will become too noisy. To reduce the number of silhouettes, a way of marking those edges that are of importance is needed. For MPM visualization the edges of more importance are those with higher discontinuities in depth. Using the magnitude of the Laplacian, edges where the discontinuity is greater than a threshold can be marked. By varying this threshold, edges corresponding to different degrees of discontinuity can be

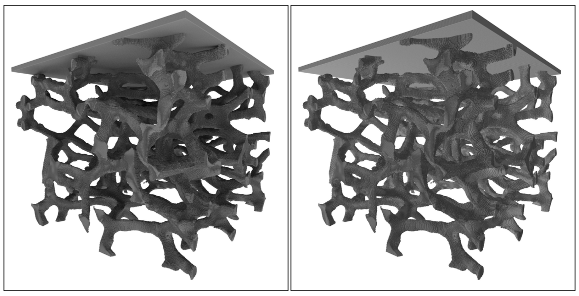

Figure 5: *Compaction of foam by a rigid plate. The foam sample is about 1 mm^3. The initial geometry was collected using micro-CT, which was then converted to a particle distribution, thus bypassing the extreme difficulties associated with constructing a suitable body-fitted mesh [BBMS06]. The left image includes ambient occlusion shading texture values. The image on the right contains direct lighting only. It took 12 hours to precompute the textures. Frame rates to render the left and right images (500x500) were 7.35 f/s and 8.67 f/s respectively. Color version (Figure 13).*

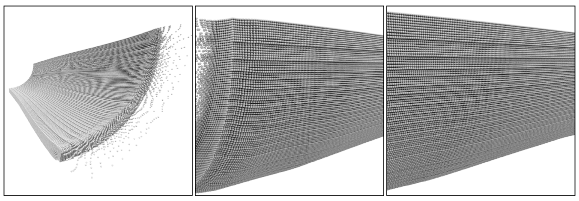

Figure 6: *Ambient occlusion shading texture values mapped on a steel cylinder, filled with detonating high explosive. Container breakup has begun. The views are identical to the ones in Figure 7. Subimages are referred to as A, B, and C, left to right. It took 66 minutes to precompute the textures. Frame rates to render image A (500x500) using direct lighting only, direct lighting with ambient occlusion, and ambient occlusion only were 18.96 f/s, 17.23 f/s, and 19.10 f/s respectively. Frame rates to render image C (500x500) were 13.89 f/s, 12.70 f/s, and 14.40 f/s.*

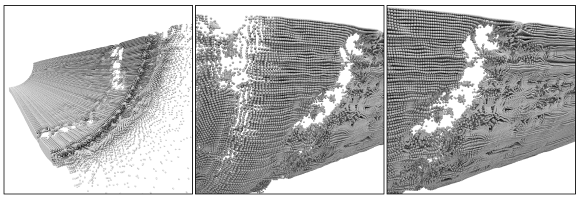

Figure 7: *Ambient occlusion shading texture values mapped on a later time step of the same simulation as seen in Figure 6. The views are identical. Subimages are referred to as A, B, and C, left to right. It took 261 minutes to precompute the textures. Frame rates to render image A (500x500) using direct lighting only, direct lighting with ambient occlusion, and ambient occlusion only were 12.00 f/s, 10.68 f/s, and 10.86 f/s respectively. Frame rates to render image C (500x500) were 9.09 f/s, 8.49 f/s, and 9.45 f/s.*

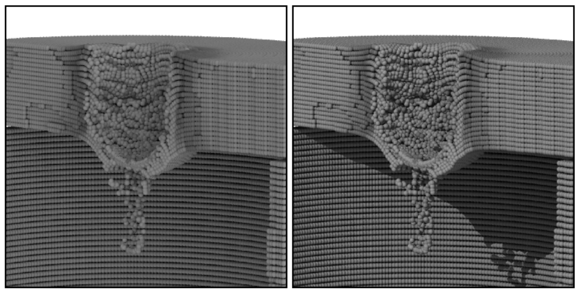

Figure 8: *Ambient occlusion texture maps are applied to the color-mapped particles. The image on the right adds direct lighting with shadows. The data set is the same as in Figure 3. Color version (Figure 14).*

selectively shown. Figure 9 shows a series of images where silhouettes are drawn for edges of decreasing discontinuity.

The threshold is insensitive to rotations or changes in the field of view as long as the relative depth remains nearly constant. If the distance between the eye point and geometry changes significantly, the threshold will need to be modified, because the relative depth values will change. User interaction, however, generally includes rotations only about the center of the object and changes in the field of view without moving the eye point closer or farther away, keeping the relative depth values meaningful to the depth threshold.

One advantage of computing imaged based silhouette edges is the lack of precomputation time. It does, however, require computation while rendering. For this particular implementation, edge detection is done in a separate thread from the image rendering allowing the computation to be overlapped with that of the rendering threads. This helps reduce the negative impact on frame rates. When computing silhouette edges, frame rates dropped by only 1-4% for scenes seen in Figures 9. The performance impact increases for scenes that rendered faster than 30 f/s. Since the silhouette computation is performed by a single thread, the frame rate is limited by the computational capabilities of that single thread. To overcome this bottleneck silhouette edge computation can be parallelized by assigning regions of the rendered image to multiple threads. For most applications with MPM data sets, however, the impact of computing silhouette edges even with only a single thread is negligible.

The impact on memory is small relative to the amount needed to store the particle data and scales with the number of pixels being rendered. Since the depth buffer is now stored, an extra four bytes per pixel is required to store the depth.

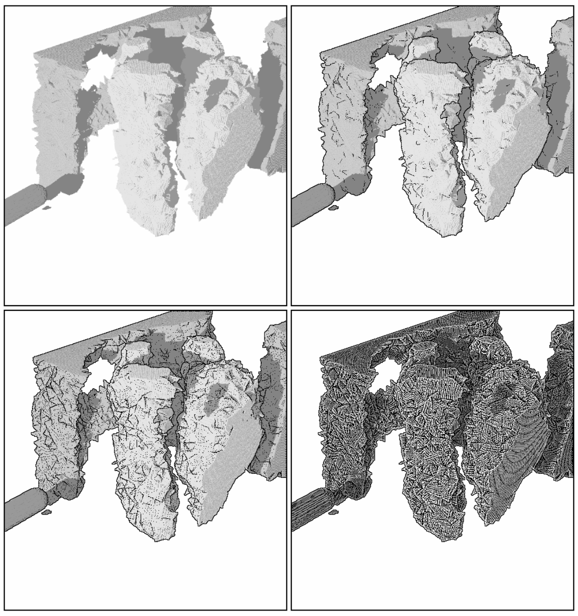

Figure 9: *By varying the threshold of the Laplacian to be counted as an edge we can change how many edges are displayed. The data visualized is a 9 mm bullet just prior to striking a model of the human torso. Several components of the torso have been removed to provide a view of the blood contained in the heart. The images are referred to as A (top left), B (top right), C (bottom left), and D (bottom right). Image A shows no silhouettes. Image B and C show progressively more and more edges. Image D shows all the detectable edges. Frame rates to render image A, B, and C at 500x500 are 2.25 f/s, 2.22 f/s, and 2.22 f/s respectively.*

4. User Experience

The ability to visualize large particle data sets in an interactive manner provides the research scientist with a set of tools that has value both in code development as well as data analysis. The Material Point Method is unique among particle methods in that it (typically) uses a Cartesian mesh as part of the solution process. This mesh can also be exploited for data visualization, by making use of techniques such as isosurfacing and volume rendering. These techniques are valuable for certain applications and audiences, but each involves a degree of compromise. On the other hand, the particles are where the data natively reside, and as such, a clear visualization of them gives the most faithful representation of the data.

The most obvious, and perhaps most often utilized, value of this capability is seen in debugging. Like any computational method, MPM is occasionally subject to ill-behaved solutions. Understanding the cause of these often requires interrogating the data where it lives, at the particles. For example, one of the primary applications of our MPM code is in the simulation of energetic devices reacting and exploding. These are complicated simulations that frequently involve several million particles, some of which represent the explosive material, while others represent the steel container in which that material is housed. If, during the course of the simulation, even *one* of those particles is accelerated to a physically unrealistic velocity during the explosion, (as occasionally happens when a particle whose mass has become too small), the progress of the simulation is essentially halted due to time stepping constraints. The methods described here allow the investigator to interact with those several million particles and find the one that is the cause of the failure. Most importantly, a clear image can be obtained of that particle's physical state, and its spatial relationship to the rest of the domain.

Of course, these simulations are frequently successful and give data that is scientifically interesting. One characteristic that was not recognized using other visualizations is the flow of the explosive material that takes place during the reaction, prior to the rupture of the container (see Figures 6, 7, and 10). Namely, the high pressure that results from the reacted explosive compresses the remaining explosive that then gives rise to instabilities and flow within that material. The relative ease one can observe the particle field evolving temporally makes such observations far more apparent than they would be using any of the other available tools. Currently this can only be done using the ray tracing system designed to visualize MPM data.

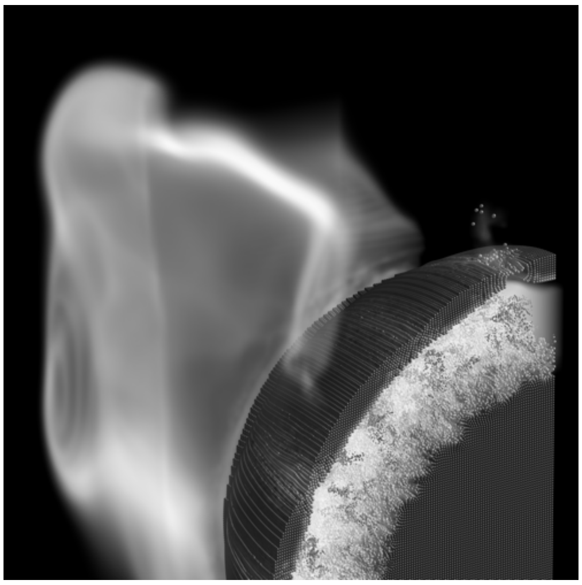

Figure 10: *A container with explosives is heated by a pool fire. The explosive starts to burn and eventually causes a rupture at the top. Color version (Figure 15).*

Ambient occlusion provides global lighting information that can be viewed interactively. This information is purely used for spatial perception. The visualization with textures using ambient occlusion was of benefit for data analysis, even without additional lighting effect. Since the visualization tool provides the ability to interactively view many time steps each containing hundred of thousands to millions of particles, having textures for all the particles in memory would be necessary. The demand on memory to store the textures reduces the number of time steps able to be loaded. The application scientists would rather be able to view more time steps than have improved lighting.

The use of silhouette edges worked well in aiding the visualization of the structures when applied to MPM particle data sets. Once briefly shown how the threshold dial worked, users were able to effortlessly tune the silhouette edges on their own. The application scientists were pleased that they could use this visualization with little impact on interactivity and more importantly no additional preprocessing. In reference to the Foam data set (see Figure 11), edge detection was particularly valuable in the analysis of this dataset. One can correlate the collisions between the individual struts that comprise the foam (as it is compressed) with small spikes in the reaction forces recorded at the boundaries of the computational domain, thus giving improved insight into the bulk material behavior. Videos of the visualization were also passed along to colleagues and used for presentation of scientific results.

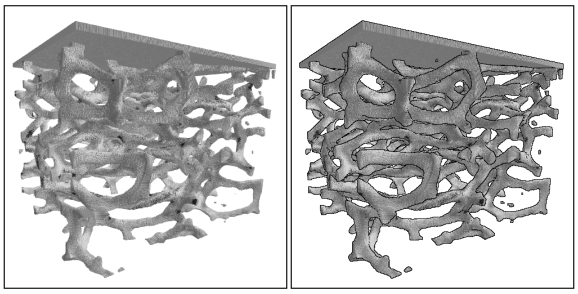

Figure 11: *A foam structure being crushed (same data as in Figure 5). The images are color mapped and shaded with direct lighting. Images on the right contain silhouette edges, the ones on the left do not. Frame rates to render the left and right images (500x500) were 8.96 f/s and 8.80 f/s respectively. Color version (Figure 16).*

5. Conclusions

Interactive visualization of large MPM data sets composed of particles is necessary for scientific analysis. To improve this analysis two enhancements were tried by the application scientists. Silhouette edges helped the scientists to better see edges of the macroscopic structure as well as address deficiencies in shadowed regions. Users were pleased with the ability to immediately make use of it without complicated interfaces.

The application scientists liked the appearance of the ambient occlusion; however, they disliked like the fact that they could only load in a fraction of the time steps they could previously. They valued being able to visualize more time steps over the ability to use ambient occlusion. The precomputation time was also a deterrent for the use of this method.

Being able to interactively explore time varying MPM data sets is crucial for understanding the results of the simulation. The ability to do this was made possible only with the availability of interactive ray tracing on large shared memory machines making this tool an integral part in the day to day exploration of MPM data.

Acknowledgments

We would like to thank the Department of Energy who funded this work through the Center for the Simulation of Accidental Fires and Explosions (C-SAFE). Data sets seen in Figures 2, 3, 4, 8, 12 and 14 are courtesy of Biswajit Banerjee of the University of Utah. We especially wish to thank Jerry Seidler and Erin Miller of the University of Washington and Scott Bardenhagen of Los Alamos National Laboratory for the foam data set found in Figures 5, 11, 13 and 16 [BBMS06]. We would also like to thank our reviewers for their thoughtful and insightful recommendations.

References

[BBMS06] BRYDON A., BARDENHAGEN S., MILLER E., SEIDLER G.: Simulation of the densification of real open–celled foam microstructures. *to appear, J. Mech. Phys. Solids* (2006).

[BBS00] BARDENHAGEN S. G., BRACKBILL J. U., SULSKY D.: The material-point method for granular mechanics. *Comput. Methods Appl. Mech. Engrg. 187* (2000), 529–541.

[BKHJ01] BRUCKSCHEN R., KUESTER F., HAMANN B., JOY K. I.: Real-time out-of-core visualization of particle traces. In *Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics* (2001), IEEE Press, pp. 45–50.

[BS00] BUCHANAN J. W., SOUSA M. C.: The edge buffer: a data structure for easy silhouette rendering. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering* (2000), ACM, ACM Press, pp. 39–42.

[GGSC98] GOOCH A., GOOCH B., SHIRLEY P., COHEN E.: A non-photorealistic lighting model for automatic technical illustration. *Proceedings of SIGGRAPH 98* (July 1998), 447–452. ISBN 0-89791-999-8. Held in Orlando, Florida.

[Goo98] GOOCH A. A.: *Interactive Non-photorealistic Technical Illustration*. Master's thesis, University of Utah, December 1998.

[GSG*99] GOOCH B., SLOAN P.-P. J., GOOCH A., SHIRLEY P., RIESENFELD R.: Interactive technical illustration. In *ACM Interactive 3D* (April 1999), ACM.

[Gum03] GUMHOLD S.: Splatting illuminated ellipsoids with depth correction. In *Proceedings of 8th International Fall Workshop on Vision, Modeling and Visualization 2003* (November 2003), pp. 245–252.

[GW01] GUILKEY J., WEISS J.: An implicit time integration strategy for use with the material point method. In *Proceedings from the First MIT Conference on Computational Fluid and Solid Mechanics* (June 2001).

[GW03] GUILKEY J. E., WEISS J. A.: Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *Int. J. Num. Meth. Eng. 57* (2003), 1323–1338.

[IKSZ03] IONES A., KRUPKIN A., SBERT M., ZHUKOV S.: Fast, realistic lighting for video games. *IEEE Computer Graphics and Applications 23*, 4 (May 2003), 54–64.

[JPH*99] JOHNSON C., PARKER S., HANSEN C., KINDLMANN G., LIVNAT Y.: Interactive simulation and visualization. *IEEE Computer 32*, 12 (Dec 1999), 59–65.

[KBHJ01] KUESTER F., BRUCKSCHEN R., HAMANN B., JOY K. I.: Visualization of particle traces in virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (2001), ACM Press, pp. 151–157.

[KPH96] KROGH M., PAINTER J., HANSEN C.: Parallel sphere rendering. In *EUROGRAPHICS Workshop on Parallel Graphics and Visualization* (Bristol, England, September 1996).

[KWTM03] KINDLMANN G., WHITAKER R., TASDIZEN T., MÖLLER T.: Curvature-based transfer functions for direct volume rendering: Methods and applications. In *IEEE Visualization* (October 2003), IEEE.

[Lan02] LANDIS H.: Production-ready global illumination. *Course 16 notes, SIGGRAPH 2002* (2002).

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (1987), ACM Press, pp. 163–169.

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Comput. Graph. Appl. 8*, 3 (1988), 29–37.

[McC01] MCCOOL M. D.: Shadow volume reconstruction from depth maps. In *ACM Transactions on Graphics* (January 2001), vol. 19(1), ACM, pp. 1–26.

[NM00] NORTHRUP J. D., MARKOSIAN L.: Artistic silhouettes: A hybrid approach. In *Proceedings of the First International Symposium on Non Photorealistic Animation and Rendering (NPAR) for Art and Entertainment* (June 2000), ACM, pp. 31–37.

[Pal99] PALMER S. E.: *Vision Science*. The MIT Press, Cambridge, Massachusetts, 1999.

[PMS*99] PARKER S., MARTIN W., SLOAN P.-P. J., SHIRLEY P., SMITS B., HANSEN C.: Interactive ray tracing. In *Symposium on Interactive 3D Graphics* (1999), pp. 119–126.

[PPL*99] PARKER S., PARKER M., LIVNAT Y., SLOAN P., HANSEN C., SHIRLEY P.: Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics 5*, 3 (July-September 1999), 238–250.

[RC99] RASKAR R., COHEN M.: Image precision silhouette edges. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), ACM, ACM Press, pp. 135–140.

[SCS94] SULSKY D., CHEN Z., SCHREYER H. L.: A particle method for history-dependent materials. *Comp. Methods Appl. Mech. Engrg. 118* (1994), 179–196.

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Computer Graphics (Proceedings of SIGGRAPH)* (August 1990), vol. 24(4), ACM, pp. 197–206.

[Ste03] STEWART A. J.: Vicinity shading for enhanced perception of volumetric data. In *IEEE Visualization* (October 2003).

[SZS95] SULSKY D., ZHOU S., SCHREYER H.: Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications 87* (1995), 236–252.

[WFG92] WANGER L. C., FERWERDA J. A., GREENBERG D. P.: Perceiving spatial relationships in computer-generated images. *IEEE Comput. Graph. Appl. 12*, 3 (1992), 44–51, 54–58.

[WPS*05] WEINSTEIN D., PARKER S., SIMPSON J., ZIMMERMAN K., JONES G.: Visualization in the scirun problem-solving environment. In *The Visualization Handbook*, Hansen C., Johnson C., (Eds.). Elsevier, 2005, pp. 615–632.

[ZIK98] ZHUKOV S., IONES A., KRONIN G.: An ambient light illumination model. In *Rendering Techniques'98* (1998), Springer-Wien, New York, pp. 45–55.

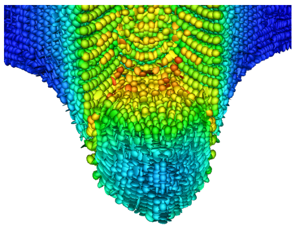

Figure 12: *Color version of Figure 2.*

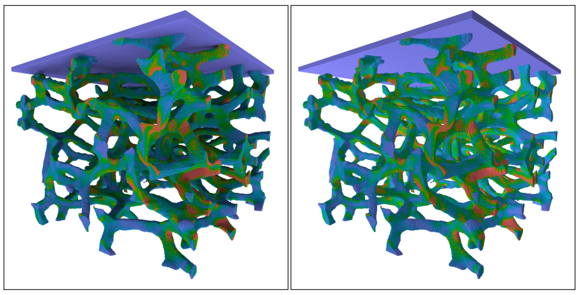

Figure 13: *Color version of Figure 5.*

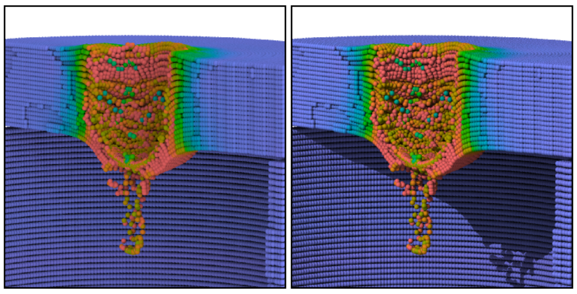

Figure 14: *Color version of Figure 8.*

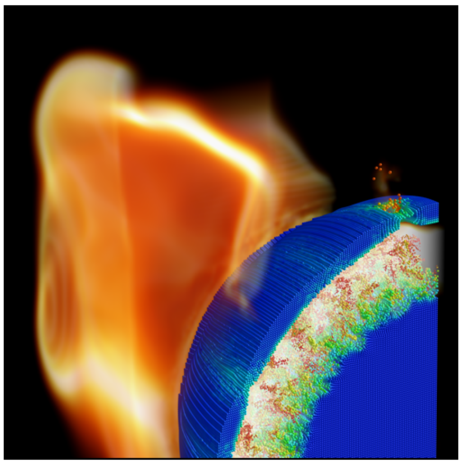

Figure 15: *Color version of Figure 10.*

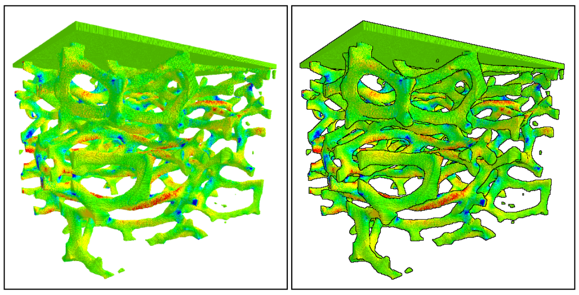

Figure 16: *Color version of Figure 11.*