Analysis of Fragmentation in Shock Physics Simulation

Kenneth Moreland, C. Charles Law, Lisa Ice, and David Karelitz

***Abstract*—Analyzing shock physics, which can involve high energies, high velocity materials, and highly variable results, is challenging. Very little can be measured during a shock physics experiment. Most experimental data is collected in the aftermath. High-fidelity simulations using codes like CTH are possible, but require a significant amount of post processing to properly understand the results. Physical structures and their accompanying data must be derived from the volumetric properties computed by the simulation. And, of course, the simulations must be validated against experiments. To capture small fragmentation effects, the CTH simulations must be run on very large scales using adaptive meshes, which further complicates the post processing. By using the scalable visualization tool ParaView coupled with customized feature identification, we are able to provide both the analysis and verification of these large-scale CTH simulations.**

I. Introduction

SIMULATION is a vital part in understanding shock physics. Although experimentation will always be the necessary tool for scientific inquiry and corroboration, the amount of data we can retrieve with simulation is limited. Experiments in shock physics usually involve high energy, high velocities, and high variability, all of which hinder detailed, accurate, and repeatable observations during the experiment. When measurements cannot be taken during the experiment, they must be taken after the experiment by observing the remaining material. Much can be learned in this manner, but the transient states during the experiment are lost.

Another limiting factor of experimentation is its high cost and slow turnaround. To create shock physics experiments, physical devices must be fabricated. These devices are then usually destroyed during the experiment. Safety and political issues also often plague shock physics experiments. In some cases, experimentation is simply not feasible.

To better analyze devices in explosive environments, Sandia National Laboratories is using the CTH shock physics analysis software [1]. In our experiments, a high fidelity mesh is important. Even the simple interactions shown in Figure 1 yield thousands of fragments, many of which are less than one microgram. Such small fragments cannot be represented if the computation grid is not fine enough to represent them. We achieve the necessary resolution by using the adaptive mesh refinement (AMR) capabilities of CTH [2] and running it on large scale computers like Sandia's Red Storm, a Cray XT3 supercomputer with over 10,000 compute nodes.

Making scientific queries from CTH simulation data requires multiple post-processing tasks [3]. First, because CTH performs volume fraction computations on an Eulerian grid, object shapes and their metrics are not immediately known. Fragments must be identified by isolating connected cells containing a material, and then statistics such as mass and volume can be derived from the shape. The second post-processing task is to validate the simulation. The results of a simulation must match the results of an experiment with an equivalent initial condition. Without this verification step, it is impossible to know if inferences from the simulation are correct. The third post-processing task is to provide tools that extract statistical information to provide better understanding of the consequences in a real-world environment.

II. Fragment Identification

The CTH AMR (adaptive mesh refinement) data sets are composed of small uniform-sampled blocks. The blocks can have various dimensions, but in any single dataset, the blocks all have identical dimensions. Blocks are commonly 12x12x12 with core dimensions 10x10x10 and an extra layer of cells which are duplicated in neighboring blocks. Although all blocks in a simulation have the same dimensions, the resolution of the blocks vary to capture different scale structures in the model. In CTH AMR data, block resolutions vary in powers of two. Other than the extra layer of ghost cells, higher resolution blocks do not overlap lower resolution blocks.

K. Moreland, L. Ice, and D. Karelitz are with Sandia National Laboratories. C. Law is with Kitware Inc.

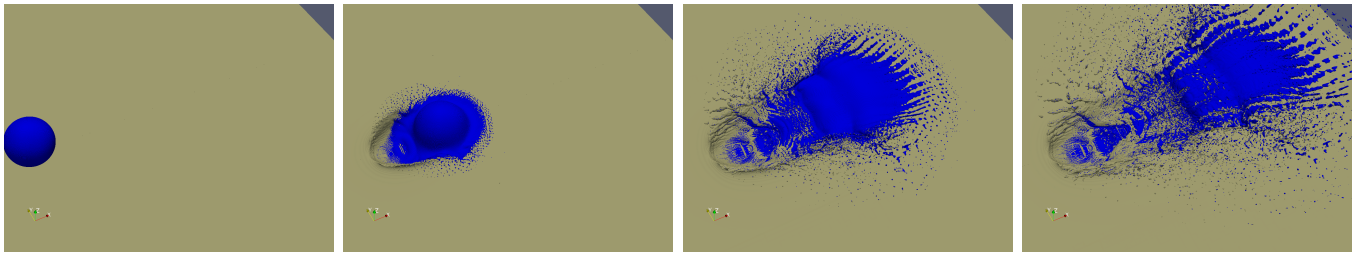

Fig. 1. Simulation of a high velocity metal ball striking a metal brick.

CTH simulations generate cell centered data and specifically create cell centered volume-fraction arrays. The volume-fraction attributes represent the fraction of the cell volume that is occupied by the represented material. There can be many materials represented, and each has a volume fraction array.

Although the basic challenge is pretty straight forward, there are several issues that make this task difficult. First: the datasets are very large and it is essential to have the data distributed across multiple processes. Second: most iso-surface algorithms work on point-centered data. Dealing with the cell-centered data adds extra complexity. Third: transitions between blocks with different levels have to be handled so that there are no cracks in the iso-surfaces. It is important that the surfaces are water tight and manifold.

A. Connectivity

The connectivity algorithm is implemented as a breadth first search considering only face neighbors of voxels. The standard approach iterates over all voxels. When an unmarked voxel above the volume fraction threshold is encountered, a new fragment id is created and the voxel is used to seed a new breadth first search. The CTH file format does not explicitly give information about neighbor relations between blocks. In order to speed the search for neighbors during connectivity a graph of blocks is created that have explicit neighbor links.

Distributed data is handled in multiple processes, each performing its own connectivity search. The first step of handling parallelism is sharing of ghost blocks so that every voxel has all neighbor voxels local in its process. Although the original CTH data contains ghost cells around each block, the supplied extra layer of cells in each block is not good enough because neighbors can have different resolutions. It is important to have the native resolution represented in the ghost cells so that transitions can be handled with no discontinuities. Rows, layers, or blocks of cells are shared between processes to get complete sets of neighboring voxels. These new blocks are added to the block graph, but are marked as special ghost blocks. After connectivity is complete on all processes, these ghost voxels are sent back to their originating processes. Fragment ids of corresponding voxels are added to a table to generate an equivalence set. This mapping is used to merge fragments that were artificially split by process boundaries. Mass properties of fragments integrated during the connectivity search are also unified at this stage of processing.

B. Fragment Surfaces

We have a few different options for creating surface models of fragments. An approach we used previously interpolates the cell volume fraction arrays at the vertices and then executes a modified marching-cubes [4] iso-surface algorithm. This strategy fails because the resulting surfaces may not correspond to the fragments extracted with face connectivity. Fragments may merge or disappear completely due to the interpolation of the volume fraction array. A viable alternative algorithm computes a dual grid [5]–[7] where every voxel becomes a point and every point becomes a 3d cell. The iso-surface is then computed directly from the volume fraction values, which have become point centered data. Although this algorithm produces water tight manifold surfaces, we selected another algorithm because it easily generates a surface in tandem with the connectivity search.

The basic idea of the algorithm is straightforward. It is equivalent to passing the input volume through a threshold filter that only keeps voxels with volume fractions greater than some threshold (Figure 2). This threshold is typically 50%. The outer surface of this set is then generated. The surface vertex positions are then adjusted to be close to the interpolated threshold value (Figure 3). This sub-voxel positioning is similar to sub-pixel positioning used for Canny edge detection [8].

We extract the surface of the fragment's voxels as they are visited during the connectivity search. When an interior voxel (root voxel) finds a face neighbor outside the fragment, the face shared by the voxels is

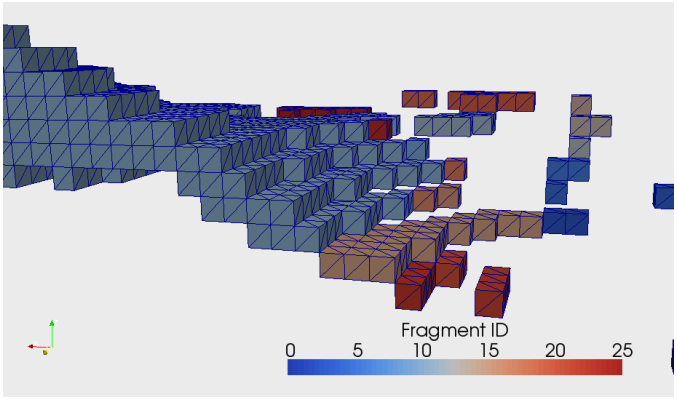

Fig. 2. Input voxels with material fractions larger than a threshold value are extracted. Face connectivity is performed to label voxels with fragment ids.

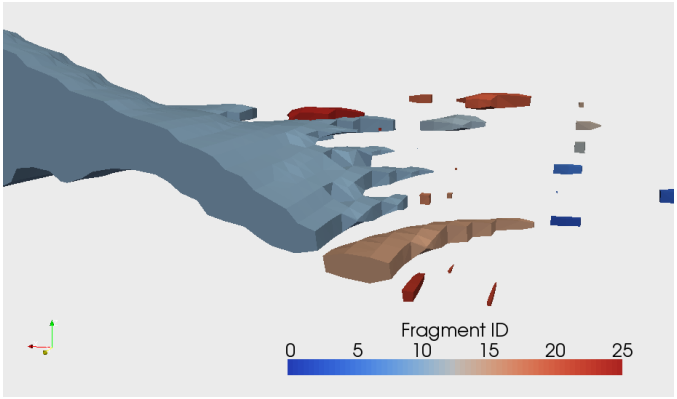

Fig. 3. Surface points are moved along the volume fraction gradient to smooth the fragment surface model. During this stage, the movement of the points is constrained to ensure the resulting surface is manifold.

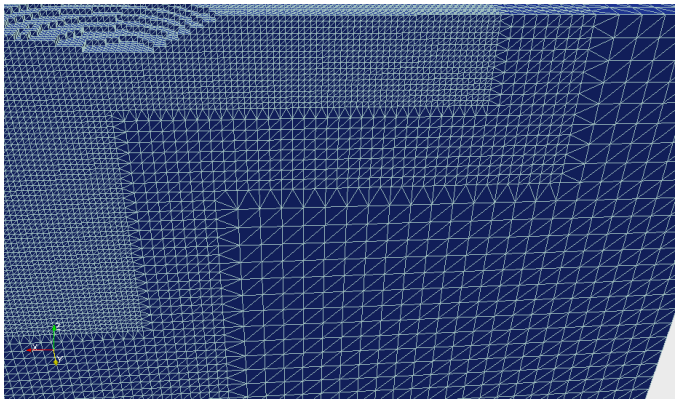

Fig. 4. Surface models of fragments are created by extracting the outer surface of fragment voxels. Care is taken at level boundaries to ensure that the fragment surface is water tight with no cracks.

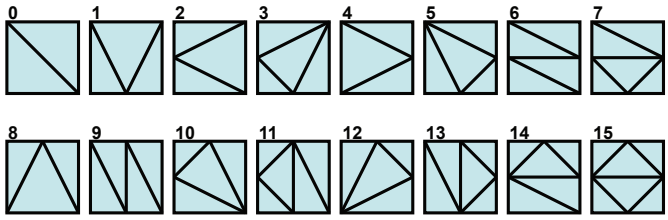

Fig. 5. These are the cases used to triangulate faces of voxels. The index into the table is computed from the four binary values indicating which of the four edges of the quad are split. Cases 6, 7, 9, 11, 13, 14 and 15 never occur in our data because our blocks are always more than one cell thick in every dimension.

triangulated and added to the fragment surface model. We also create a surface for voxel faces that have no neighbors. This creates capping surfaces for fragments touching the boundary of the dataset. Faces are not generated for ghost cells so surface models from multiple processes fit together nicely.

Care is taken when triangulating faces to avoid T-junctions when adjacent cells are from different levels (Figure 4). When we generate a face, we find all voxels that touch the face. We use these voxels to determine which edges of the face must have a middle point to mesh with neighboring faces of a higher resolution. We use a 16 entry case table to triangulate the faces (Figure 5). Each edge contributes one bit to the case index. There is no attempt to merge points from neighboring faces. Duplicate points are created and they are merged in a separate post processing phase.

Sub-voxel positioning is used to smooth the blocky surface into a more accurate representation of the fragment (Figure 3). The points are moved to be close to the interpolated threshold value and are also positioned so the final surface will be manifold. For every point we find the eight neighbor voxels that surround the point. We threshold the eight volume fractions and use a case table to mask the resulting values. The mask removes the influence from voxels that are not directly connected (through the eight neighbor voxels) to the root. This masking keeps the surface from pinching down opposite surfaces to non-manifold points or edges. We take the gradient of the masked values to determine which direction we should move the point. We normalize the vector so that the largest component is 1 unit. The resulting vector has 26 discrete possible values that correspond to the faces, edges and corners of a cube. This vector is the direction we use for the point displacement. This computed direction is like the normal of the unsmoothed surface at the point. This computation of direction keeps the mesh well behaved and avoids the surface folding back on itself.

To compute the magnitude of the displacement we assume all eight voxels have the same resolution as the root voxel. We limit the magnitude of the displacement to be half the size of the root voxel. We choose the

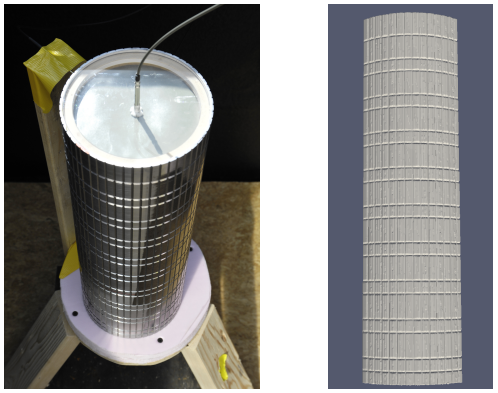

Fig. 6. A test article (left) and corresponding quarter-symmetry CTH model of the outer case (right).

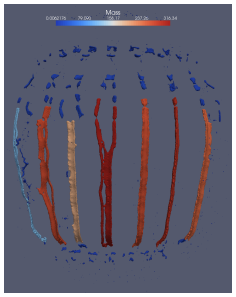

Fig. 7. Initial quarter-symmetry CTH simulation at 2.0×10^{-4} seconds colored by fragment mass.

displacement so that the tri-linear interpolated volume fraction at the new point is as close to the threshold as possible.

III. Simulation Verification

One of the most important tasks we can perform with the classified fragments is to validate the simulation. In simulation validation, an experiment is performed, and a simulation is run with its initial conditions set equivalent to the experiment setup. The measurements taken during the experiment are then compared with the results of the simulation. Figure 6 shows one of the test articles we constructed and detonated and the corresponding initial CTH model used for simulation, which can be used to analyse the fracturing behavior of the material [9].

The intention of the grooves in the test article are to control its fragmentation when it detonates. We would expect the fragments to follow the scores so that they would have similar width and height. Instead, the simulation, shown in Figure 7, clearly shows the cylinder fragmenting into long vertical strips.

This fragmentation is corroborated by the experimental data. Figure 8 shows the witness plate that was placed within proximity of the experiment's exploding cylinder. The long indentations from the damage clearly indicate

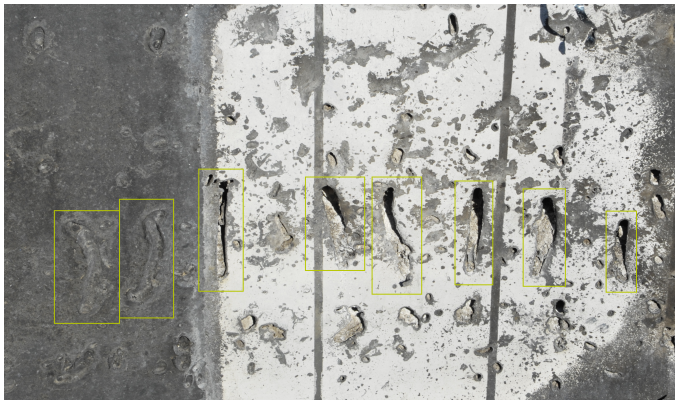

Fig. 8. Witness plate from the test article. Notice the fragment strip indentations highlighted in yellow.

Fig. 9. Representative fragments from the test article.

long thin fragments oriented in the vertical direction. Fragments collected after the experiment, shown in Figure 9, are also in the same shape and size as those computed by the simulation.

In an attempt to better understand how to control fragmentation with the grooves, an additional simulation was run with an initial model containing deeper grooves. The resulting simulation, shown in Figure 10, suggests that the cylinder will now fragment in smaller pieces as expected. This result is not yet verified by experiment, but gives good information on how to design the experiment.

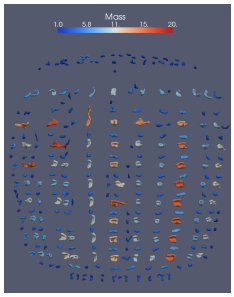

Fig. 10. A CTH simulation starting with an initial model having deeper grooves at 2.0×10^{-4} seconds. The color map has a different range than that shown in Figure 7.

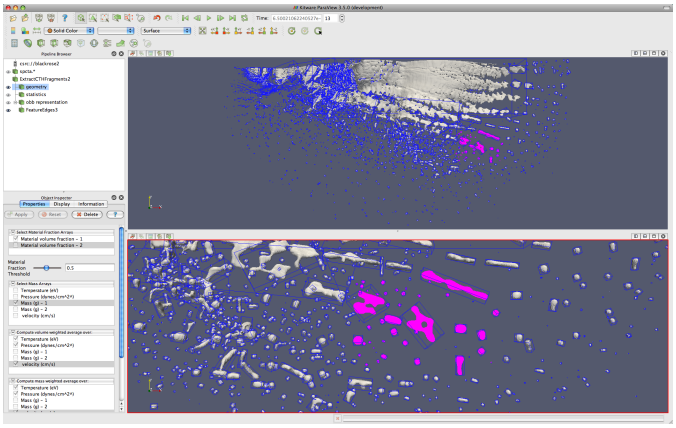

Fig. 11. A large CTH simulation containing about 180 million cells with its fragments identified and characterized by bounding boxes. A set of fragments has been selected visually.

IV. Data Queries

Once the fragments of the simulation are successfully identified, deriving physical characteristics of the fragments is straightforward. For instance, by fitting an oriented bounding box around each fragment, as shown in Figure 11, we can provide some general information and compute statistics about the fragment shapes. We can also readily compute the mass and volume of each fragment. If the simulation produces other field information, such as temperature, pressure, or velocity, then we can determine the weighted average for each fragment based on the mass or volume.

ParaView allows you to visually explore, select, and inspect the fragments. However, when the fragmentation is complex and the number of fragments is high, extracting information from the data as a whole can be tedious or impossible. Often a better approach is to use the statistics made available by the fragment identification. For example, Figure 12 shows alternatively the heaviest and lightest fragments. Because the largest and smallest fragments differ by six orders of magnitude and because the distribution of fragment size is heavily biased to the smaller fragments, it is greatly beneficial to group fragments based on mass before deriving statistical information.

ParaView's histogram filter is also an important tool for analyzing the distribution of data. Figure 12 uses a histogram to show the distribution of mass for the smallest fragments. One possible use for this distribution is to compare the data from the simulation to other simulations or to experimental data. Due to the highly variable nature of the physics, it is impossible to directly compare individual fragments between simulations or experiments. However, distributions are sometimes a reliable way to compare these types of data.

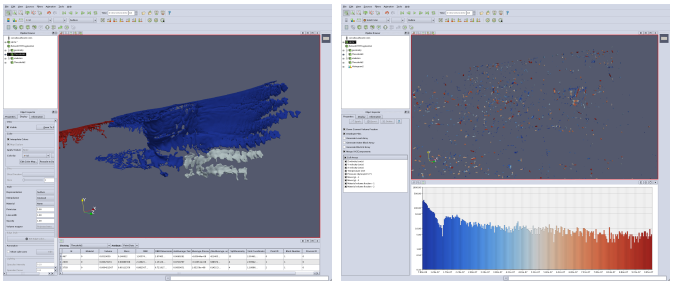

Fig. 12. On the left the three heaviest fragments are identified. On the right are the lightest fragments (those with mass between 10^{-8} and 10^{-6} grams) with a histogram showing the distribution of the weight.

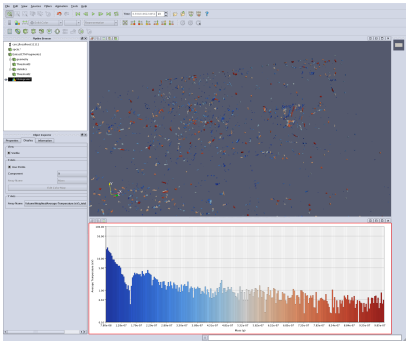

Fig. 13. Plot of mass versus average temperature.

Another use of the histogram tool is provide to comparisons between two variables. In addition to simply counting up the instances of fragments in each bin, the histogram filter is capable of computing the average field value of all fragments in the bin. Figure 13 uses the histogram filter to show the relationship between mass and temperature. The data shows that the smallest fragments tend to have the highest temperature.

A frequent quandary of these simulations is to estimate the effect these exploding materials can have on nearby objects. To determine the time and location fragments might strike an object, we provide a query tool that allows us to intersect the fragments with a plane at any time step. We can view the cross section of the fragment/plane intersection as shown in Figure 14. The intersections can be annotated with any of the material properties calculated for the fragments as previously mentioned.

In addition to the qualitative information about the shape provided by the intersection, we can also extract quantitative information. By creating a single mark for each fragment (shown in Figure 15), we can provide quantitative information about the center of each intersection as well as the properties of each intersecting fragment including volume, mass, and velocity.

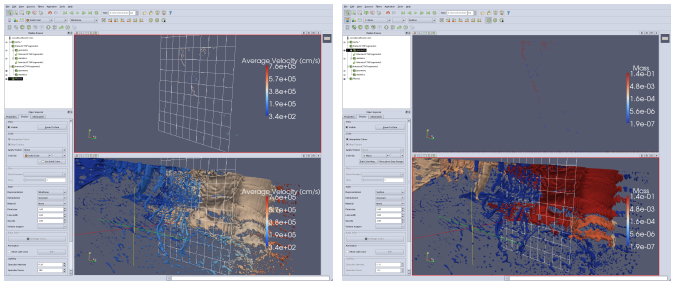

Fig. 14. Fragment/plane intersections. The top images show the fragment cross sections at the intersection. The bottom images show all fragments with respect to the plane. The left images are annotated by velocity and the right by mass.

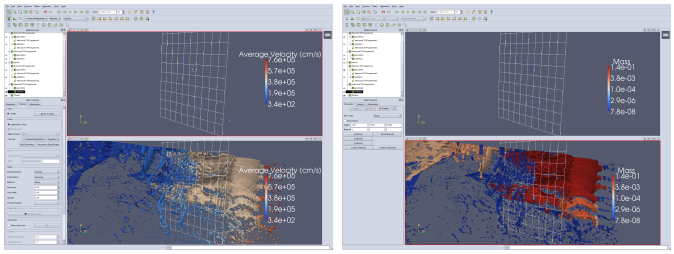

Fig. 15. Fragment/plane intersections. To top images show a mark at the center of intersection for each fragment. The bottom images show all fragments with respect to the plane. The left images are annotated by velocity and the right by mass.

V. Future Work

One frequent problem encountered when simulating shock physics is the tracking of features at multiple scales. Although it is straightforward for a simulation to give an accurate picture of the overall effects of an explosion, the details of the effects of individual fragments are lost. The fidelity required to accurately portray, for example, how an individual fragment affects another object it impacts cannot be achieved when representing the environment as a whole.

We can get around this problem by running a subsequent simulation modeling only a single fragment but at a much higher resolution. We do this by first extracting the polygonal representation of the fragment and its material properties, which is already done with the algorithm discussed previously, and using this to create an input mesh for a subsequent simulation. For the most part, this already works. We are currently working on one final issue caused by the nature of the extraction of fragment surfaces from volume fractions; nearby surfaces sometimes touch and share vertices, which breaks the manifold preconditions of the meshing algorithm. We are finishing the ability to detect and correct these pinches of the data and expect to be able to perform the subsequent meshing and simulation soon.

Another project we are working on is the integration

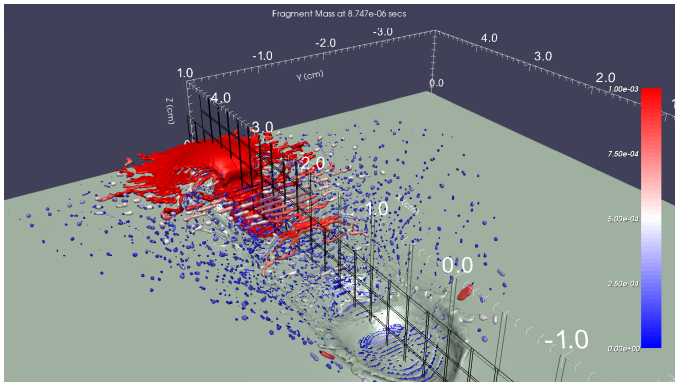

Fig. 16. An image captured from within a running CTH simulation using our fragment identification module.

of the fragment tracking into a running CTH simulation. Our approach is similar to that of Clarke and Mark [10]; we create code that uses CTH's Spymaster interface to access data generated in a running CTH simulation while it is still in core. Our code converts the data into VTK data object types and uses our algorithms to identify fragments and extract surfaces and statistical information. Figure 16 shows an example image generated during a running CTH simulation that relies on fragment identification.

The advantage of running or processing *in-situ* with the simulation is that we remove the file I/O bottleneck. When processing data after the simulation has finished, we are limited to the data written to spyplot files. These files are large and grow proportionally with the size of the simulation job. Typically, data in spyplot files are written only sporadically. Writing out all of the data produced during a large CTH simulation is prohibitive in terms of both time and disk usage. As we move to petascale computing and beyond, we predict that the fraction of data that can be stored to disk will get ever smaller.

However, when the post processing is running with the simulation, we potentially have access to all the data produced with minimal overhead. Furthermore, the resulting data from the post processing, such as images, statistical values, and polygonal surfaces, are usually much smaller than the input from which they are derived. Thus, in-situ processing allows us to generate results with a much higher fidelity.

VI. Acknowledgments

This work was done in part at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National

Nuclear Security Administration under contract DE-AC04-94AL85000.

REFERENCES

[1] E. S. Hertel, R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. Mcglaun, S. V. Petney, S. A. Silling, P. A. Taylor, and L. Yarrington, "CTH: A software family for multi-dimensional shock physics analysis," in *Proceedings of the 19th International Symposium on Shock Waves*, 1993, pp. 377–382.

[2] R. G. Schmitt, D. A. Crawford, R. L. Bell, and E. S. Hertel, "Adaptive mesh refinement and multi-phase flow in the CTH," in *Workshop on Numerical methods for multi-material fluid flows*, Paris, France, September 2002.

[3] D. B. Karelitz, L. Ice, J. Wilke, S. W. Attaway, and K. D. Moreland, "Post-processing V&V level II ASC milestone (2843) results," Sandia National Laboratories, Tech. Rep. SAND2008-6183, 2008.

[4] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics*, vol. 21, no. 4, pp. 163–169, July 1987.

[5] S. Schaefer and J. Warren, "Dual marching cubes: Primal contouring of dual grids," in *Proceedings of Pacific Graphics 2004*, 2004, pp. 70–76.

[6] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, vol. 21, no. 3, pp. 339–346, July 2002.

[7] G. M. Nielson, "Dual marching cubes," in *Proceedings IEEE Visualization 2004*, October 2004, pp. 489–496.

[8] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, November 1986.

[9] L. C. Chhabildas, T. F. Thornhill, W. D. Reinhart, M. E. Kipp, D. R. Reedal, L. T. Wilson, and D. E. Grady, "Fracture resistant properties of aermet steels," *International Journal of Impact Engineering*, vol. 26, pp. 77–91, 2001.

[10] J. A. Clarke and E. R. Mark, "Extending post-processing and runtime capabilities of the CTH shock physics code," in *Proceedings of the Users Group Conference (DOD-UGC'05)*, 2005, pp. 300–303, dOI=10.1109/DODUGC.2005.29.